2020-12

DEPLOYING AN ICS HONEYPOT IN A CLOUD COMPUTING ENVIRONMENT AND COMPARATIVELY ANALYZING RESULTS AGAINST PHYSICAL NETWORK DEPLOYMENT

Bieker, Matthew C.; Pilkington, Darry

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DEPLOYING AN ICS HONEYPOT IN A CLOUD COMPUTING ENVIRONMENT AND COMPARATIVELY ANALYZING RESULTS AGAINST PHYSICAL NETWORK DEPLOYMENT

by

Matthew C. Bieker and Darry Pilkington

December 2020

Thesis Advisor: Neil C. Rowe
Co-Advisor: Thuy D. Nguyen

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2020	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE DEPLOYING AN ICS HONEYPOT IN A CLOUD COMPUTING ENVIRONMENT AND COMPARATIVELY ANALYZING RESULTS AGAINST PHYSICAL NETWORK DEPLOYMENT	**5. FUNDING NUMBERS** W0A60
6. AUTHOR(S) Matthew C. Bieker and Darry Pilkington	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Monterey	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Industrial control systems (ICSs) provide important services in national critical infrastructure but are increasingly the subject of cyberattacks. The need for ease of maintenance and operational convenience encourages using cloud services, increasing their security vulnerabilities, and knowing what threats to expect that would help in defending cloud-based ICSs. This thesis tested an ICS honeypot (decoy system) called GridPot that was deployed in a third-party cloud environment and simulated a microgrid distribution system. We compared data from a GridPot instance deployed on an in-house server with three cloud-deployed GridPot instances with varying configurations. Overall results showed that the cloud-deployed GridPots had comparable traffic to the non-cloud GridPot, but it yielded less ICS-specific traffic, though what occurred appeared more deliberate. Nearly all attacks on the cloud-deployed GridPots showed little sophistication about ICS protocols. Our results further confirmed that cloud-based honeypot owners must maintain awareness of cloud service providers that recycle IP addresses to avoid exploits on previously used IP addresses. We conclude that ICS honeypots in the cloud are an effective tool for collecting cyberattack intelligence, and they do not appear to discourage attacks by being in the cloud.

14. SUBJECT TERMS industrial control systems, cloud computing, honeypot	**15. NUMBER OF PAGES** 75
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

DEPLOYING AN ICS HONEYPOT IN A CLOUD COMPUTING ENVIRONMENT AND COMPARATIVELY ANALYZING RESULTS AGAINST PHYSICAL NETWORK DEPLOYMENT

Matthew C. Bieker
Lieutenant Commander, United States Navy
BS, University of Maryland University College, 2016

Darry Pilkington
Lieutenant, United States Navy
BS, Trident University International, 2013

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN CYBER SYSTEMS AND OPERATIONS

and

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2020

Approved by: Neil C. Rowe
Advisor

Thuy D. Nguyen
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

Alex Bordetsky
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Industrial control systems (ICSs) provide important services in national critical infrastructure but are increasingly the subject of cyberattacks. The need for ease of maintenance and operational convenience encourages using cloud services, increasing their security vulnerabilities, and knowing what threats to expect that would help in defending cloud-based ICSs. This thesis tested an ICS honeypot (decoy system) called GridPot that was deployed in a third-party cloud environment and simulated a microgrid distribution system. We compared data from a GridPot instance deployed on an in-house server with three cloud-deployed GridPot instances with varying configurations. Overall results showed that the cloud-deployed GridPots had comparable traffic to the non-cloud GridPot, but it yielded less ICS-specific traffic, though what occurred appeared more deliberate. Nearly all attacks on the cloud-deployed GridPots showed little sophistication about ICS protocols. Our results further confirmed that cloud-based honeypot owners must maintain awareness of cloud service providers that recycle IP addresses to avoid exploits on previously used IP addresses. We conclude that ICS honeypots in the cloud are an effective tool for collecting cyberattack intelligence, and they do not appear to discourage attacks by being in the cloud.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

APCI	Application Protocol Control Information
ASDU	Application Service Data Unit
DCS	distributed control system
HMI	human-machine interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
ICS	industrial control System
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IT	information technology
OT	operation technology
PaaS	Platform as a Service
PCAP	packet capture
PLC	programmable logic controller
RTU	remote terminal unit
SNMP	Simple Network Management Protocol
SaaS	Software as a Service
TCP	Transmission Control Protocol
SCADA	supervisory control and data acquisition
SHODAN	sentient hyper-optimized data access network
XML	Extensible Markup Language

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This thesis would not have been possible without the help of several people. Thank you to our cohort for the support, friendship, and study sessions which were essential to our success here at Naval Postgraduate School. Thank you to the professors and staff at Naval Postgraduate School for the education, which has provided us the foundational knowledge necessary for our thesis. Thank you to Jeff Dougherty who was incredibly helpful in adapting his research into our own.

We also thank Professors Rowe and Nguyen who pushed and guided us through this process. We are grateful for your direction and most of all your patience as we took on a topic that was outside our comfort zone. To stay the course and finish the thesis, especially during a global pandemic, gives us an incredible sense of accomplishment.

Matt would like to thank his wonderful, youthful, and beautiful wife Deana, who put up with his grumpiness over the last nine months; Jim the Dog, for the moments of levity; and the rest of his family, especially his brother, Tommy. I love you, and miss you every day. Finally, thanks to Darry for taking this journey with me; I could not have done this without you.

Darry would like to thank his wife, Mari, for her patience and support during the thesis process. How you are able to handle three children (Drew, Milli, and myself), I will never know. Thank you, Drew and Milli, for making the home something fun and exciting to be confined to during these unusual times. I also want to thank Matt: you took the words out of my mouth; I could not have done this without you.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This thesis addresses the defense of industrial control systems (ICSs) for the National Cyber Strategy. The work developed methods to identify threats to ICSs by operating honeypots (decoys) in a cloud network and analyzing their attack data. We analyzed the collected data to determine whether a cloud deployment discouraged attackers, whether it was more resilient to attacks than a traditional network, and whether it could support analysis of attacks. The Navy relies on critical infrastructure; these results can help protect this infrastructure and aid in decision-making for systems used by the Navy in defense platforms, and shipyards in support of SECNAVINST 3501.1D.

A. INDUSTRIAL CONTROL SYSTEMS AND THE CLOUD

ICSs are used in many applications, most notably for critical infrastructure such as water-distribution networks, electricity generation and distribution, oil refineries, nuclear plants, building management, and public transportation systems (Tesfahun & Bhaskari, 2016). Because of their original physical isolation as well as their logical isolation from digital networks, security was not initially difficult for these systems (Antón et al., 2017). Today, ICSs rely on widespread TCP/IP (transmission control protocol/internet protocol) network products rather than the proprietary protocols and specialized products (hardware and software) used previously (Stouffer et al., 2015), this change has reduced system downtime and simplified configuration (Centre For The Protection of National Infrastructure, 2010). Although remote network access to control systems that monitor or alter processes is convenient, it opens ICSs to many Internet-based attacks (Antón et al., 2017). In the last few years, industries have started integrating cloud-computing services into ICSs. Cloud computing provides on-demand access to a shared pool of configurable computing resources including networks, servers, software, and services (Combs, 2020). With cloud-based systems, operating costs are reduced. The growth in ICSs, including supervisory control and data acquisition (SCADA) systems responsible for remotely accessing local control modules for operational control and data collection, has encouraged interest in adopting cloud technology. Market research estimated that in 2019 the global

SCADA market size exceeded 30 billion U.S. dollars and will grow over 7.5% in the next six years, attributing most growth to the increasing adoption of cloud-based SCADA and infrastructural development (Bhutani & Wadhwani, 2020). An example of the trend is Vipond Controls, which has developed iSCADA, a remote SCADA system with a local interface. iSCADA is available as a service on Vipond's server cloud and allows customers to access virtual machines that can monitor and control assets remotely, including changing controllers and troubleshooting processes (Combs, 2020). iSCADA was used by an oil and natural gas exploration company to bring 300 wells online in a single month (Gavin, 2018).

To further assess the adoption of cloud services for industrial control systems, we used the Sentient Hyper-Optimized Data Access Network (SHODAN), a Web-based tool that discovers and classifies devices connected to the Internet (Shodan, n.d.). We searched it for IP addresses associated with TCP port 2404, which is uniquely associated with the Telecontrol Equipment and Systems protocol IEC 104 (Matousek, 2017), the protocol of focus for our thesis. Searching just the cloud servers of DigitalOcean, the provider used for this thesis, it found 237 addresses worldwide. Many more probably exist under other cloud servers.

B. ATTACKS ON INDUSTRIAL CONTROL SYSTEMS

ICSs have long been targeted by attackers. In 2006, a remote hacker installed malicious software on a water filtration system in Harrisburg, Pennsylvania, severely affecting plant operations (National Service Center for Environmental Publications [NSCEP], 2012). In 2015, the BlackEnergy malware campaign compromised the systems of three energy distribution companies in Ukraine, by seizing control of SCADA systems, remotely shutting down substations, disabling infrastructure components, and temporarily disrupting power to over 225,000 customers (Beach-Westmoreland & Stycynski, 2019). In 2016, the Ukraine electric grid was attacked again with malware called Crashoverride, exploiting the IEC104 protocol, targeting Internet-connected human-machine interfaces; it deenergized a Kiev transmission substation, causing outages (Dragos Inc., 2017). In December 2019, an ICS of the Bahrain Oil Company Bapco was attacked with malware called DUSTMAN, which was designed to overwrite data (Kaspersky, 2020). The attacker exploited a remote execution vulnerability on a virtual private network, harvested

credentials, and distributed its malware throughout the network. In December 2019, ICS ransomware called EKANS (snake backward) was discovered in a commercial malware repository (Dragos Inc., 2020). Besides encrypting files as is typical in ransomware, EKANS stopped specific ICS processes on a "kill-list" including data-historian clients as well as user interfaces and possibly control systems.

Honeypots are sites deliberately designed to be attacked. ICS honeypots can operate to look like a real ICS to lure attackers from production networks or to collect attack methods without exposing real network resources (Gjermundrød & Dionysiou, 2015). The cloud environment permits easier construction of honeypots by an economy of scale; users can configure multiple virtual machines for honeypots from a pool of IP addresses and run them on the same local network. Cloud services can allow users to place honeypots in different geographical locations to see more diversity of traffic (Chandra & Madhuri, 2012, p. 6).

However, moving ICSs to a cloud environment increases their vulnerability. ICS cloud services are exposed to attack protocols and are not designed for security but have been used in control systems for decades with few software updates (Antón et al., 2017). These factors, combined with easily available exploitation tools and reconnaissance services like SHODAN, make it easy to quickly exploit existing vulnerabilities to attack ICS cloud systems. On the other hand, that means that ICS honeypots can collect much useful intelligence about attacks.

C. THESIS OUTLINE

Chapter II examines previous work on cloud configurations, power grid systems, ICS architecture, and honeypots. Chapter III describes the architecture of our cloud-based ICS honeypot. It also describes microgrid power systems, Conpot and GridPot architectures and protocols, and the GridLAB-D power-system simulator. Chapter IV describes our research methodology, giving configuration specifics for cloud deployment of each honeypot, alterations made to the default templates to make our honeypots less obvious, and data collection and network analysis methods. Chapter V shows and discusses the results. Chapter VI states conclusions.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND AND RELATED WORK

A. CLOUD DELIVERY MODELS

Three delivery models for a cloud environment are Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS) (Bokhari et al., 2016, pp. 890–892). IaaS uses virtual machines only for hardware resources like servers, networks, and operating systems; the client manages the operating system, data, and software. SaaS, also called cloud application services, runs applications remotely; a license is provided to the client to access the software. PaaS provides basic services such as databases and programming tools remotely and is primarily used for program development.

Three primary cloud configurations are private, public, and hybrid (Tinankoria & Babak, 2017). Private cloud services are hosted at an owning organization and maintained by them. Public cloud services are hosted off-site, usually by a commercial entity. Hybrid cloud services interconnect local resources with off-site resources.

Cloud services provide multi-tenancy, scalability, and elasticity for all delivery model levels (Lehrig et al., 2015, p.83). Multi-tenancy allows the sharing of the same computing services among many users (Morsy et al., 2010). Scalability permits the allocation of existing resources to maintain performance when the workload increases. It is particularly useful for service models where clients have many processes like honeypots operating simultaneously. Elasticity permits rapid expansion and contraction as necessary of computing capacity from the cloud service provider, matching the current demand with the minimum required resources. It helps systems handling bursts of traffic, or a customer minimizing the computing power they require for their processes. Cloud systems can also provide off-site emergency data back-ups, data-analytics, development, data storage, and system monitoring and control (Piggin, 2015). All these features help ICSs.

Several security issues occur with cloud implementations. Moving outdated legacy systems to cloud-based architectures exposes vulnerabilities to a large audience. It also requires relying on service providers as well to safeguard proprietary information, and vulnerabilities in the service provider's infrastructure can cause system compromise

outside of the client's control (Gonzales et al., 2017, p.523). The security of a cloud system depends on installed security applications, sharing permissions between tenants, hypervisor protection measures, and the level of protection provided to the user by the cloud service provider, which makes security more complex than with a standalone application. The National Institute for Standards and Technology (NIST) provides standards related to requirements of accessibility, interoperability, performance, portability, and security of cloud services (National Institute of Standards and Testing [NIST], 2013), but not all cloud services follow them.

B. INDUSTRIAL CONTROL SYSTEMS

ICSs combine control components to run infrastructure processes. Two common types are distributed control systems (DCS) and supervisory control and data acquisition (SCADA) systems (Stouffer et al., 2015). DCSs control automated production systems within a geographic area; often they integrate with corporate networks so management can observe production. SCADA systems oversee controllers of a production cycle of a localized process. Feedback or feed-forward control loops maintain processes around a desired "set point." Figure 1 shows a typical configuration of a remotely operated SCADA system. Programmable logic controllers (PLCs) are often used in DCS and SCADA systems to locally manage a process through feedback-control loops. They can also operate partly independently as they have programmable memory to execute instructions. PLCs generally do closed-loop functions.

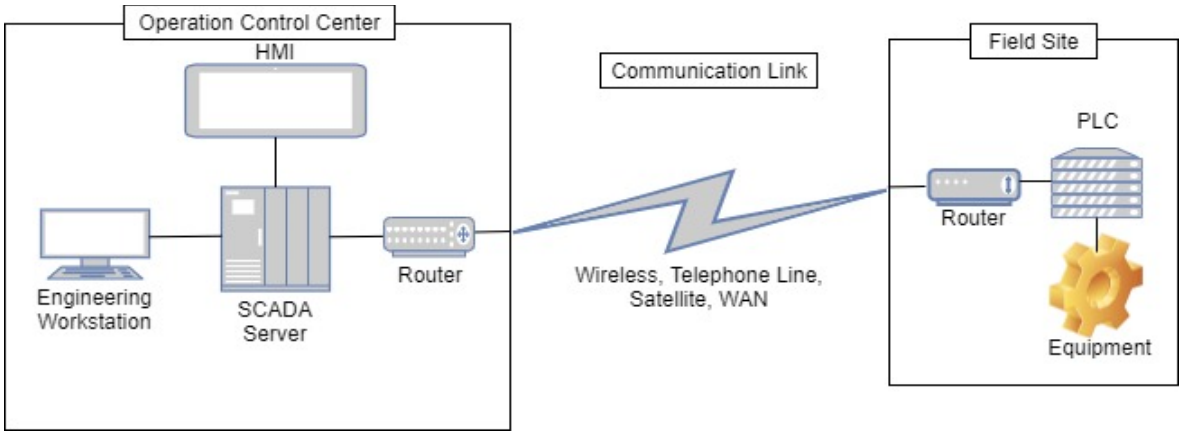

Figure 1. A typical SCADA configuration

A power grid also called an electric-utility network, sends electrical power from power plants to users (American Public Power Association [APPA], n.d.). This process has three phases: generation, transmission, and distribution. Electricity is created by generation, then the voltage is stepped up in voltage by a transformer for transmission. Distribution steps down the transmitted voltage and distributes it to users through distribution lines and transformers. Our thesis simulates a 13-house power-grid distribution system that communicates using the IEC 104 protocol.

C. HONEYPOTS

Honeypots detect reconnaissance attempts, analyze traffic, and deflect attacks from real networks to collect information about attack methods. This is done by using software that imitates the behavior of the desired target system. Honeypots have no production value and are solely intended to encourage reconnaissance and attack activity (Redwood et al., 2015). Honeypots can be virtualized or physical (Provos, 2008). Using many honeypots on a network with different configurations will attract different kinds of attackers. Figure 2 shows a typical honeypot network with several honeypots alongside a real production system.

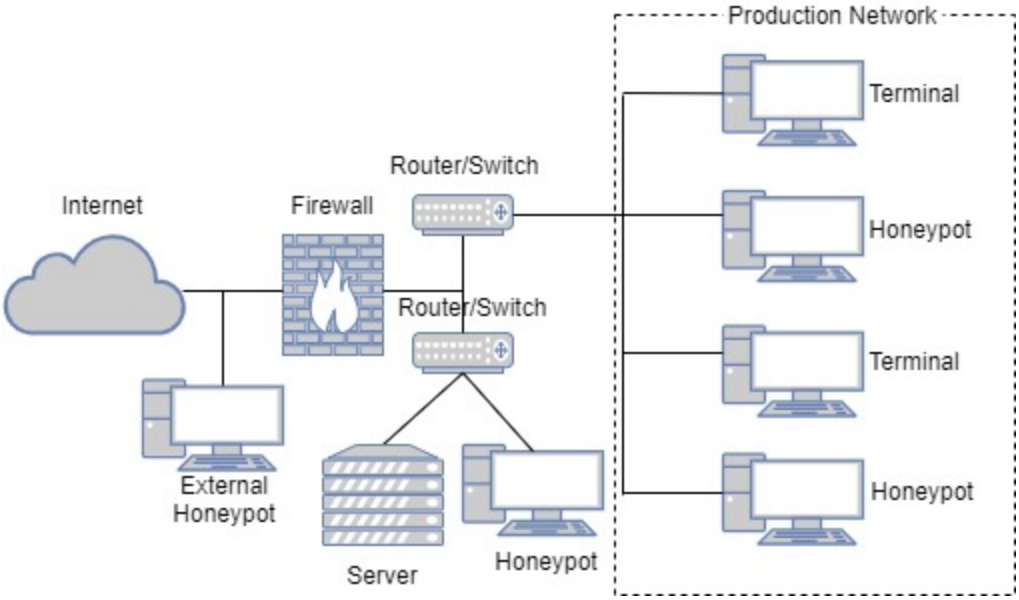

Figure 2. A typical honeypot deployment alongside a production system

Honeypots can be high-interaction, medium-interaction, and low-interaction. Low-interaction honeypots are less resource-intensive and allow the attacker to interact only with limited simulated services and static data. Medium-interaction honeypots provide some feedback to commands issued by an attacker but offer limited applications and resources to interact with; they allow a better understanding of attack methods than low-interaction honeypots. High-interaction honeypots allow intruders to access a nearly complete set of applications and resources on a system. A high-interaction honeypot will be more responsive and can entice attackers to spend more time interacting with it (Khoshnaw, 2017). For our thesis, we used the medium-interaction honeypot GridPot (Sk4ld, 2015) to emulate a power grid system using the IEC 60870-5-104 protocol (IEC 104) for device-to-device communications (Matousek, 2017) and the Hypertext Transfer Protocol (HTTP) for Web client communications. GridPot did not provide an interactive Web-based user control interface but did provide an HTTP system status page accessible by Web-based browsers, and it responded to IEC 104 protocol messages when they were sent to port 2404 of the cloud server.

D. PREVIOUS ICS HONEYPOTS

In this section we discuss several previous ICS honeypot projects.

One honeypot project ran a large-scale low-interaction ICS honeypot using the Amazon EC2 cloud service for 28 days (Serbanescu et al., 2015, pp.21-25). It serviced several protocols including Modbus, the Distributed Network Protocol (DNP3), the Inter-Control Center Communications Protocol (ICCP), the ICS protocol IEC 104, the Simple Network Management Protocol (SNMP) (v1/2/3), the Trivial File Transfer Protocol (TFTP), and the Extensible Messaging and Presence Protocol (XMPP). Interactions were either connection attempts or protocol-based requests. Modbus interactions were 15% of the interactions and IEC 104 interactions were .002%. SNMP had the most interactions with 85% of traffic. The honeypot did not respond to IEC 104 traffic. The researchers observed that being indexed by SHODAN increased reconnaissance attempts.

Another SCADA honeypot tried to entice attackers to use high-value or new attacks on SCADA ICS systems (Redwood et al., 2015). It was a realistic high-interaction honeypot that modified the Conpot honeypot (Conpot, n.d.). They used the IEC 61850 protocol as a wrapper for a less secure Modbus protocol called GOOSE and the Manufacturing Message Specification (MMS) protocol communicating between devices. Attacker activity came through a Web-based user interface on port 80. The honeypot also interfaced to Pacific Northwest National Laboratory's electrical-power simulator GridLAB-D (Gridlabd.org, 2020) that allowed attackers to think they could manipulate voltage flow and other electrical settings. The honeypot appeared to be effective at enticing attackers to interact using high value or zero-day exploits.

Another honeypot project simulated the operations of electrical companies (Barak, 2020). It mimicked an electrical provider's network with an information-technology environment including a data network, an operational-technology environment operating physical processes and machinery, and a controller interface. Soon after launch, it was subject to ransomware attacks. They were done by exhaustively guessing the username and password of a remote administration interface used for troubleshooting. The attacker executed a PowerShell script to create a backdoor to maintain persistence, and several

exploits were uploaded, including credential-stealing tools that allowed attackers to move through the network. The same project launched a similar honeypot later, and it was compromised too (Barak, 2020). Attackers installed backdoors by creating additional user accounts, in preparation for selling access to it. Malware was also installed including crypto-miners, and the honeypot was used for phishing and distributed denial-of-service attempts.

Q-GridPot is a commercially available network-appliance honeypot designed to model electric-power grids (Quantalytics, 2019). It uses GridPot to create a fake power-grid infrastructure. It can also create customizable operator interfaces with a realistic appearance. It comes with network intrusion-detection systems, packet-capture tools, and forensic software, as well as tools to defend against attacks.

A project at NPS ran GridPot with GridLAB-D simulation software (Kendrick & Rucker, 2019). It collected data for 19 days, exposing the protocols HTTP, S7 Communications, SNMP, and Modbus to outside traffic. SHODAN correctly identified it as a honeypot when it used site history but could not otherwise identify it. It attracted a wide array of traffic, totaling 1,525,059 packets and 165 megabytes of data, much of which was scanning and attack-related. This research showed that attackers will exploit Web-based SCADA vulnerabilities on an externally facing connection.

Another project at NPS used the Conpot honeypot to find indications of compromise (Hyun, 2018). It parsed log data to identify protocols being attacked and countries of origin as well as the content of attack packets. It analyzed logs of the eight default protocols provided by Conpot: EtherNet/IP, the Communication Industrial Protocol (CIP), HTTP, S7 communications, the Intelligent Platform Management Interface (IPMI), SNMP, the Building Automation and Control Networking Protocol (BACnet), and Modbus. HTTP was the primary protocol seen, with Modbus being second and being repeatedly targeted by a small pool of IP addresses on specific days. S7 Communications was the third most used protocol, and BACnet, IPMI, SNMP, and EtherNet/IP were used significantly less. Conpot appeared to be a good way to extract indicators of compromise, even though it is a low-interaction honeypot.

III. METHODOLOGY AND DESIGN

This chapter discusses the design and construction of our experiment. We also describe the cloud hosting of our honeypots, and how we deployed and operated them. More specific details of our configuration and implementation are in Chapter 4.

A. OVERALL FRAMEWORK

Our experiment used a virtual machine containing an instance of GridPot created by Dougherty (Dougherty, 2020) and deployed it three ways in a cloud environment. The goal was to compare data of the deployment methods for IEC 104 and HTTP traffic. We wished to determine differences in the quantity, type, or source location of traffic.

B. DIGITALOCEAN CLOUD ENVIRONMENT

Our experiment needed an IaaS cloud service with adequate resources to run different GridPot instances as well as the Conpot and GridLAB-D applications. We considered Amazon Web Service (AWS) and DigitalOcean (DO) as potential platforms. Campus information-technology specialists told us that Amazon Web Service discourages the running of security applications such as penetration testing and honeypots, and DigitalOcean was cheaper for the same services, so we chose the latter.

The DigitalOcean cloud platform provided us with control over resources (memory, processing power, and storage), a choice of the operating system, snapshots and regular backups, remote console access, firewall control, and options for international deployment (Digital Ocean, 2020b). DigitalOcean provides a virtual server platform which they call a "droplet." We used three droplets to host three GridPot instances. Of the four droplet plans available (basic, general-purpose, CPU-optimized, and memory-optimized), we chose a general-purpose droplet plan which gave us increased processing power on a dedicated CPU, more memory, and ample storage (Digital Ocean, 2020a). This permitted copying Dougherty's Ethernet-based honeypot to a virtualized cloud environment. The ability to create snapshots and regular backups allowed the quick restoration of a droplet should it crash or become compromised. We deployed and operated the honeypots remotely with

"droplet consoles." Each droplet has a built-in DigitalOcean firewall to control inbound and outbound traffic. DigitalOcean has international servers in North America, Asia, and Europe, we used the North America and Asia servers for our experiments.

C. CONPOT

GridPot requires Conpot, a low-interaction ICS honeypot that uses an internal IEC 104 server, which interfaced with GridLAB-D, a simulator of an electrical grid (Conpot, n.d.). Conpot provides templates for several protocols and services including IEC 104, GridPot, Kamstrup_32, and Guardian_AST. Although IEC 104 was available in Conpot, we used the GridPot template for it because it was used by Dougherty. Our experiments only collected data on HTTP and IEC 104 traffic.

D. GRIDPOT

GridPot is a two-layer honeypot that simulates an ICS for an electrical grid (Redwood et al., 2015). The GridPot implementation in our experiment included Conpot and GridLAB-D (Figure 3). Conpot controlled the honeypot layer while GridLAB-D controlled the electrical-grid modeling. GridPot is freely available on-line. Our experiment used an earlier version of GridPot that had been modified for Dougherty's work. Client access to the GridPot came in through the Conpot on ports 80 and 2404. Changes to the simulated data were sent to GridLAB-D simulation and status changes were returned to the client through Conpot.

E. DOUGHERTY'S WORK

Our GridPot implementation came from the work of (Dougherty, 2020) and his Phase 1 in particular. GridPot provided limited attacker interaction on an ICS protocol interface, which Dougherty's work called "low fidelity." Therefore, Dougherty created an additional layer of obscurity by having GridPot communicate with GridLAB-D, which would emulate an ICS system. The intent was to create a more interactive honeypot, which attackers could find more attractive, thus creating more traffic for data collection.

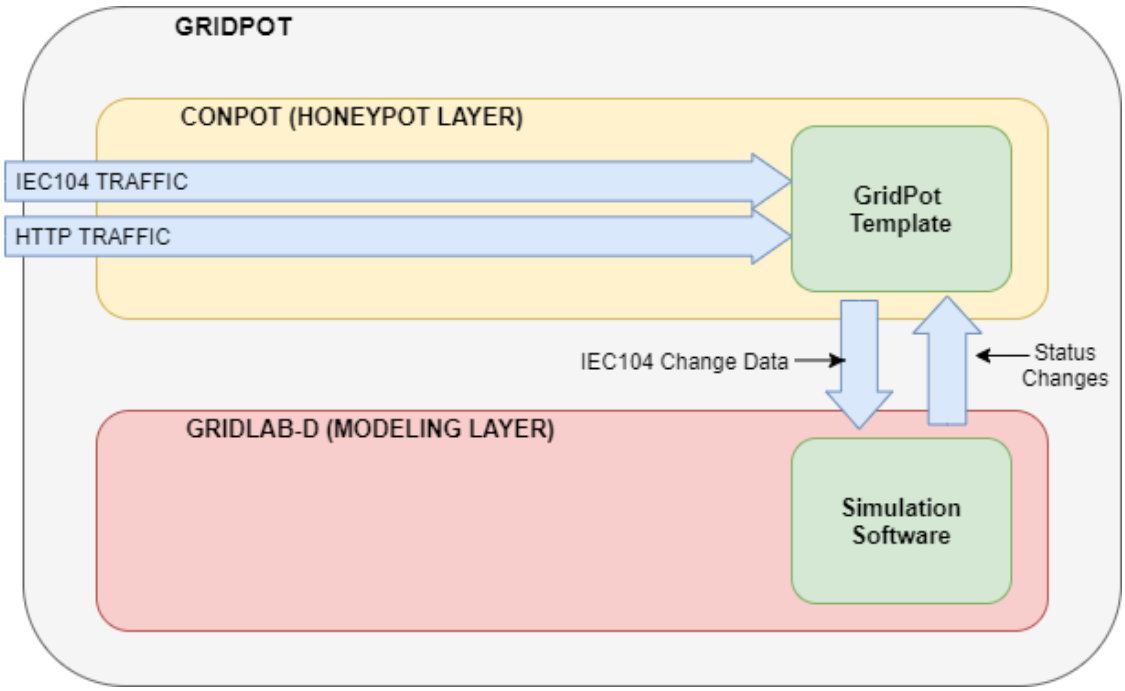

Figure 3. Basic GridPot architecture

F. GRIDLAB-D

GridPot uses GridLAB-D, open-source software for simulating power distribution, that receives IEC 104 change requests from Conpot to modify the simulation parameters, and returns updated simulation data to Conpot (Gridlabd.org, 2020). GridLAB-D contains over 40 simulation models ranging from water heaters to wind turbines. Our GridPot implementation used the IEEE 13-Node Model with Houses simulation (Figure 4), It emulated a microgrid, an isolated electric grid for power transmission and distribution (Lantero, 2014). The model emulated a microgrid with switches, voltage regulators, power transformers, and end-users.

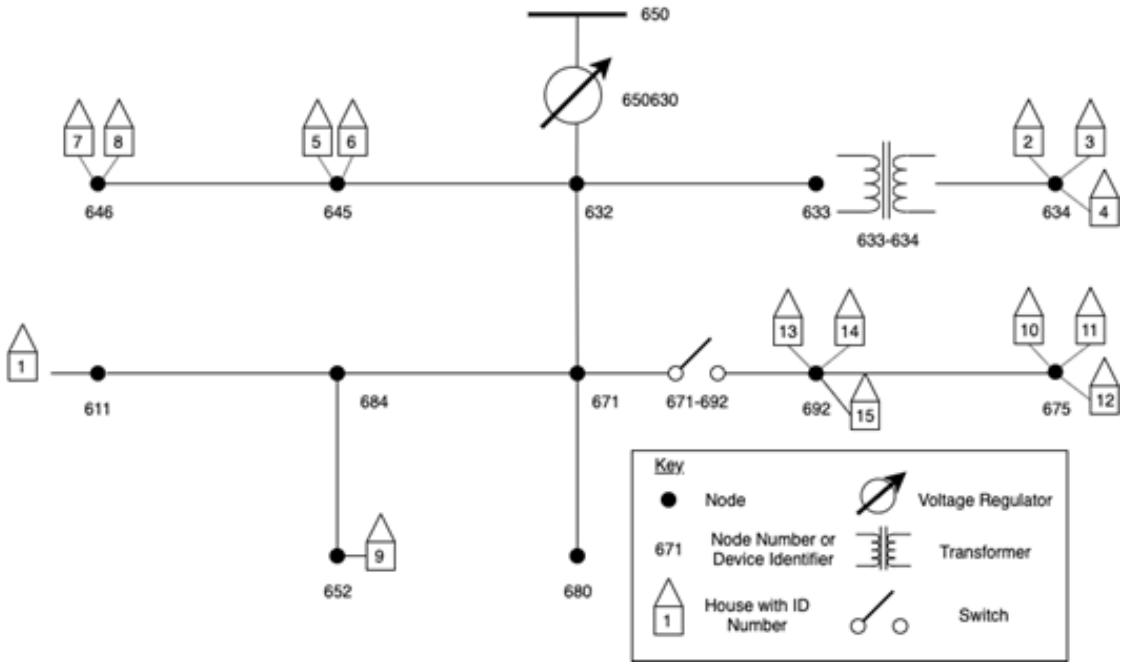

Figure 4. The IEEE 13-node model with houses. Source: Dougherty (2020)

G. RELEVANT PROTOCOLS

1. HTTP

Most traffic we saw with our honeypots was of the Hypertext Transfer Protocol (HTTP). It is an application-level protocol that communicates between Web-based clients and Web servers through TCP port 80 (Fielding et al., 1999). A client uses different HTTP methods to request various resources from the server. The HTTP methods seen in our results were OPTIONS, GET, HEAD, POST, PROPFIND AND CONNECT. OPTION returns communications options for a specific resource. GET returns a specified resource. HEAD is like GET except that its response contains information about the specified resource, not the actual resource. POST sends input from the client to the server. PROPFIND retrieves properties from the resource. (Dusseault, 2007). CONNECT establishes a tunnel between the client and the server resource.

2. IEC 60870-5-104

IEC 60870-5-104 is an application-level protocol, often called IEC 104 for short, for ICS communication through port 2404 (Matousek, 2017). IEC 104 remotely controls

equipment and systems with three types of Application Protocol Control Information (APCI) frames, called format frames in this thesis: I-format, S-format, and U-format. I-format frames transfer information including instructions between a controlling station and a controlled device, and contain an important subpart called an Application Service Data Unit (ASDU) (Figure 5). S-format frames execute supervisory functions from a controlling station to the controlled device and acknowledgments in return. U-format frames execute control functions from a controlling station to a controlled station and contain an ASDU. Herein, these three frame types are also referred to as I-frame, S-frame, and U-frame. In our experiments, the IEC 104 server in Conpot received I-format and U-format frames. Data was then transferred from the server to and from GridLAB-D for simulation control and monitoring (Figure 3). Dougherty's work allowed the attacker to make detailed manipulation within the IEC 104 protocol, and our experiments took it further by changing the environment and implementation methods.

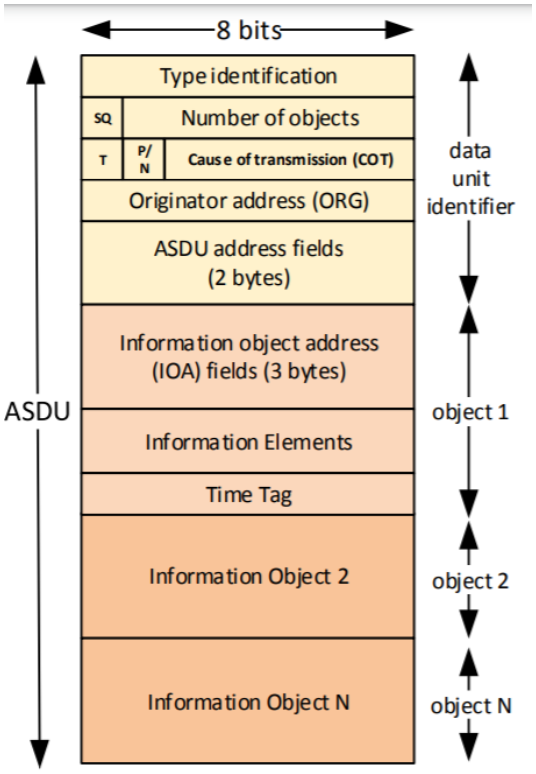

Figure 5. ASDU structure. Source: Matousek (2017)

THIS PAGE INTENTIONALLY LEFT BLANK

IV. EXPERIMENT IMPLEMENTATION

This chapter discusses experiment configuration and implementation, data collection, and data analysis. The research had three experiments with different cloud implementations as described in Section IV.A. The data analysis methods are described in Section IV.C. The results of our analysis are in Chapter V.

A. CONFIGURATION AND IMPLEMENTATION

We conducted three experiments with different deployment methods (Figure 6). We compared the data from these experiments with Dougherty's data as indicated by a dashed line. The droplet resources, operating system, and software were consistent throughout the experiments. Each of the three environments was created in a DigitalOcean "droplet" or virtual machine (DigitalOcean, 2020a) using DigitalOcean's pre-built Ubuntu Linux 18.04 (LTS) x64 image.

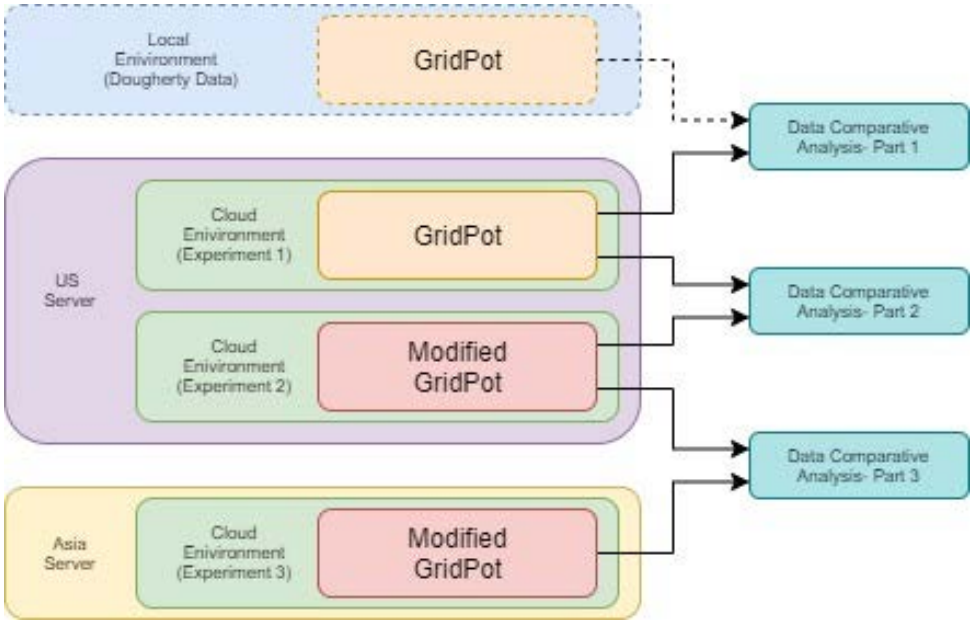

Figure 6. Experiment structure

All three experiments used DigitalOcean general-purpose droplets with a dedicated CPU, 16GB memory, and 50GB solid-state disk (SSD) storage. Each droplet had a single unique outward-facing IP address. We accessed droplets with a DigitalOcean console at its website and by using secure-shell protocol (SSH) from local devices. During testing, a DigitalOcean firewall blocked inbound traffic to the GridPot process. During live implementation, the firewall permitted inbound traffic on SSH port 22, HTTP port 80, and IEC 104 port 2404.

Experiment 1 used Dougherty's Phase 1 deployment (Dougherty, 2020) as modified slightly for a cloud environment. The purpose of Experiment 1 was to compare traffic between a physical deployment and a cloud deployment. While installation required additional dependencies for cloud operation as described in Section IV.D, the GridPot code was the same as in Dougherty's. Before Experiment 1, we confirmed the SSH, HTTP, and IEC 104 ports were open by a network-mapper (Nmap) scan of its IP address.

Experiment 2 was like Experiment 1 except for several code changes to data constants within the template files that were intended to better simulate power-grid components and make it less obvious as a honeypot. We again confirmed that the SSH, HTTP, and IEC 104 ports were open using Nmap and we also confirmed that port 2404 had valid IEC 104 responses. We ran test interactions to confirm that the HTTP responses correctly reflected the modified code.

Experiment 3 was like Experiment 2 except for being deployed in Asia. We performed similar confirmations on ports before deployment.

B. DATA COLLECTION

Data was collected from 17 August 2020 to 03 September 2020 (17 days) for Experiment 1 and from 16 September 2020 to 04 October 2020 (18 Days) for Experiments 2 and 3, which were run simultaneously. Data was collected using packet-capture (PCAP) software and Conpot logs.

Besides our packet captures, which collected complete packets on all HTTP and IEC 104 traffic, we got packet captures from Dougherty's Phase 1 research with Wireshark

software (Wireshark · Go Deep., n.d.). His data was collected for just HTTP and IEC 104. We used Tshark, a terminal-oriented version of Wireshark (D.2. Tshark: Terminal-Based Wireshark, n.d.). Packet capturing was configured to roll over into a new file every 25 MBs, but this created difficulty for transfer to repositories. Some repositories had a maximum file upload size of 10 MBs; therefore, we recommend limiting the file size to less than 10 MBs.

Conpot logs were also collected. They are text files that log and timestamp honeypot-layer activity from when Conpot is initialized until it is either brought down or crashes. They include session starts, session timeouts, HTTP responses, IP addresses, client details, I-format frame data, and U-format frame data. Conpot logs can also be used as a backup if packet data is missing for a period. An example Conpot log is given in Figure 7. It shows the initialization of the Conpot and the first HTTP session established.

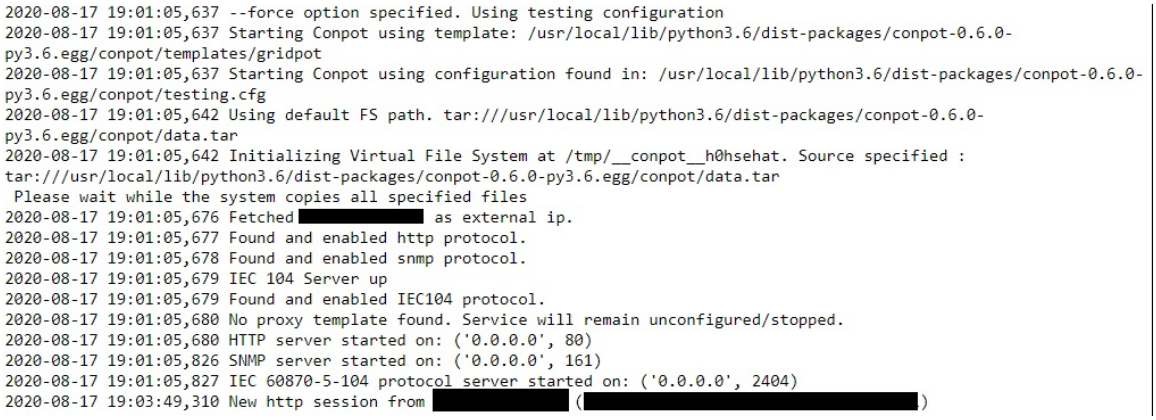

Figure 7. Conpot log example

C. DATA ANALYSIS

For each experiment, all packet captures were combined into a single file in time order. A Python parser, written by Dougherty, extracted relevant data and stored it in data frames. This included information about HTTP sessions, HTTP method requests, IEC 104 sessions, IEC 104 messages, IEC 104 errors, and IP geolocation information. We then used Jupyter Notebook to calculate statistical data for our analysis. Another parser counted HTTP/IEC 104 sessions, HTTP/IEC104 method types, unique IP addresses, IP location

data, and IEC 104 format frame data in the packet captures. Conpot logs were used in our analysis to recognize I-format and U-format frame traffic from Shodan and other probes.

D. PROBLEMS ENCOUNTERED

The initial installation of GridPot on the cloud server encountered several build errors caused by missing Python packages. After installing the required packages, we were able to successfully create a GridPot instance on the cloud server. The missing Python packages were: pytest, pluggy, text-unicode, appdirs, ipaddress, and greenery.

During Experiment 1, a Virtual Network Computing (VNC) desktop environment ran on top of the Ubuntu Linux 18.04 (LTS) operating system for our convenience. This environment let us remotely log in to the server and open several terminal instances at a time. However, in Experiment 1 the VNC interface stopped working, and we had to use the default terminal console provided by the operating system. When invoked from the console, the *bash* command to open additional terminal instances did not work, so we executed our programs as background processes. This also required us to switch our packet capture software to Tshark as Wireshark required a desktop environment for installation.

When we initialized GridPot, Conpot, and Tshark as background processes through a secure shell connection, they were suspended when the secure-shell session ended, which was undesirable because a honeypot must work continuously. As a work-around, Conpot, GridPot, and Tshark were initialized from the droplet console on DigitalOcean. The secure shell was only used for system monitoring, data collection, and data transfer.

During Experiments 2 and 3, Nmap IEC 104 probe results initially did not reflect the honeypot code changes we made. We found that Conpot needed to be re-initialized for the code changes to take effect. Re-initialization required running Conpot's setup script (setup.py), which created the corresponding runtime image and placed it in the build library. That image was used when initializing Conpot.

A goal of our experiments was to fool Shodan into believing that our experiments were real ICS systems. To measure this, we used the Shodan command line interface tool Honeyscore, which uses data from known honeypots to create criteria to identify additional

honeypots. Our Experiments 2 and 3 returned a score of 0.0 stating they were not honeypots However, we discovered that Honeyscore had been developed for a specific project and had not been updated recently and should not be used as a measure of honeypot detectability, most of the code has now been rolled into the regular honeypot detection mechanism used on SHODAN's website (J. Matherly, personal communication, October 21, 2020).

E. CONPOT SETUP

For an initial test, we installed a Conpot on a virtual machine using VirtualBox and Ubuntu 18.04. We then successfully created a cloud version on DigitalOcean to develop skills with installation methods in the cloud using DigitalOcean. Thanks to the success of Dougherty's GridPot research, we abandoned the Conpot setup and created the more powerful cloud-based GridPot implementation described earlier.

THIS PAGE INTENTIONALLY LEFT BLANK

V. ANALYSIS

This chapter discusses our observations from our cloud-based experiments.

Table 1 shows traffic statistics collected for all experiments used during our analysis.

Table 1. Overall comparison of traffic for all experiments

	Dougherty Phase 1	*Experiment 1*	*Experiment 2*	*Experiment 3*
Number of days	157	17	18	18
Number of unique countries	126	67	66	72
Number of unique IP addresses	2820	541	652	3750
Total number of requests	27021	5564	3728	28509
Total HTTP requests	26368	5376	3698	28485
Total IEC 104 messages	653	188	28	24
Total IEC 104 malformed messages	557	171	13	14
Total IEC 104 valid messages	96	17	17	10
Mean HTTP requests per day	239.714	298.672	194.634	1499.21
Min HTTP requests per day	1	51	27	815
Max HTTP requests per day	2191	2438	1123	2820
Mean IEC 104 messages per day	10.532	10.444	2.55	2.667
Min IEC 104 messages per day	1	7	1	1
Max IEC 104 messages per day	130	16	6	9
Mean valid IEC 104 requests per day	2.341	2.429	3.0	2.0
Minvalid IEC 104 messages per day	1	1	1	1
Maxvalid IEC 104 messages per day	8	6	4	3

Experiment 1 was indexed by SHODAN within the first day of operation, showing ports 22, 80, 2404, and 5901 as open and correctly identifying it as an ICS honeypot. SHODAN only reported ports 22 and 80 as open for Experiments 2 and 3, possibly due to historical indexing data. Experiment 3 received IEC 104 messages from an IP address associated with SHODAN, but port 2404 was not reported as open on SHODAN's website during the experiment. Figure 8 shows Experiment 1 daily HTTP traffic, IEC 104 traffic, and valid IEC 104 message counts, and is representative of traffic received during the other experiments.

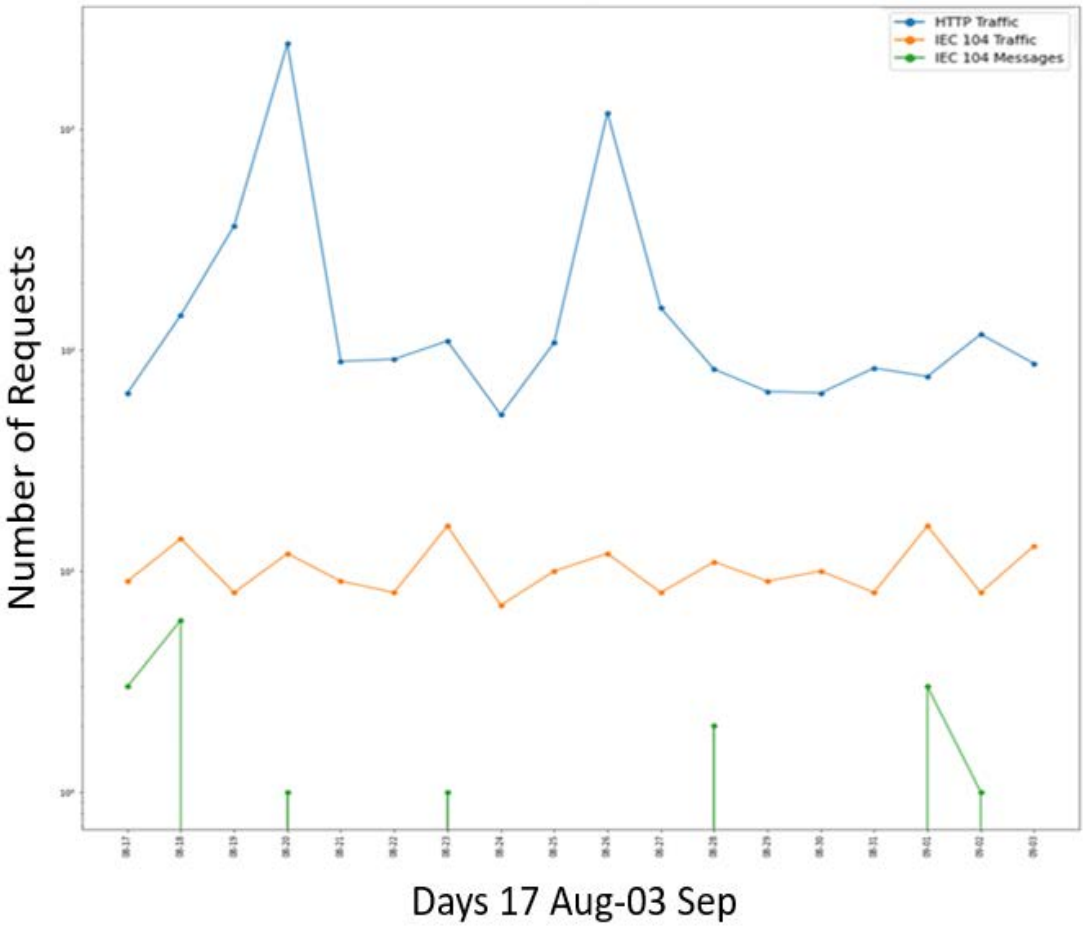

Figure 8. Experiment 1 HTTP and ICS daily traffic

A. SESSION DATA

Sessions were defined as all packets exchanged by a socket (site) pair on a given date (as with Dougherty). We counted the number of addresses that established a single HTTP-only session, those that established multiple HTTP sessions, those that established a single ICS (IEC 104) session, those that established multiple ICS sessions, and those that established both HTTP and ICS sessions (see Table 2). The total number of sessions in this table does not equal the count in "HTTP only sessions" plus the count in "Multiple HTTP sessions" because each IP address could have multiple HTTP sessions, and we only record the number of IP addresses establishing multiple sessions.

Table 2. Session data

	Dougherty Phase 1	*Our Experiment 1*	*Our Experiment 2*	*Our Experiment 3*
Days	157	17	18	18
Total Number of Addresses Seen	2820	541	652	3750
Number of HTTP-Only Sessions	2747	506	640	3740
Number of Multiple HTTP Sessions	548	118	127	2464
Number of ICS-Only Sessions	53	4	9	8
Number of Multiple-ICS Sessions	32	1	3	3
Number of Both HTTP and ICS Sessions	20	31	3	3
Total Number of Sessions	7121	1811	1515	21474

1. More About Sessions

Figure 9 compares the number of IP addresses that connected to our honeypots for a single HTTP-only session, multiple HTTP-only sessions, and those that connected to the ICS IEC 104 server at least once ("Other").

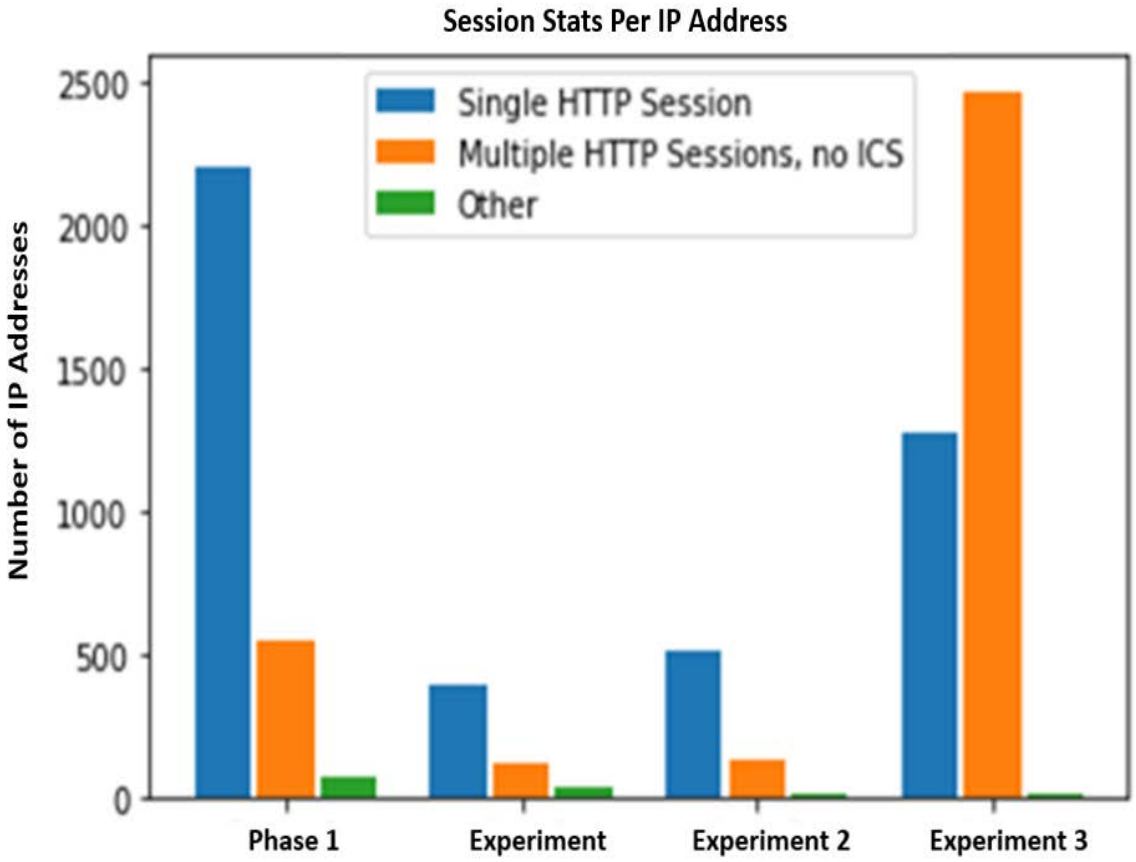

Figure 9. Session statistics per IP address

Table 3 shows the reported source country distribution of HTTP sessions. Unsurprisingly, the United States was in the top three of all experiments, and China and Russia appeared in the top ten of US-based experiments while Russia did not appear in the Asia-based experiment and China was seventh.

Table 3. Distribution of top 10 countries with HTTP sessions

	Dougherty Phase 1		*Experiment 1*		*Experiment 2*		*Experiment 3*	
	Country	*Percentage of Total*	*Country*	*Percentage of Total*	*Country*	*Percentage of Total*	*Country*	*Percentage of Total*
1	China	22.5	United States	40.7	United States	34.6	United States	40.9
2	United States	16.5	Russia	16	China	16.5	Singapore	13.5
3	Russia	12.3	China	9.1	Russia	10.7	Canada	4.9
4	Taiwan	4.7	Germany	6.4	Netherlands	6.4	Netherlands	4.6
5	Brazil	3.8	Netherlands	3.2	United Kingdom	5.4	France	4.5
6	Netherlands	3.6	Brazil	2.6	India	3	Germany	3.4
7	Germany	3	Romania	2.4	Brazil	2.3	China	2.3
8	Japan	1.8	Seychelles	2.2	Germany	1.9	Italy	2.1
9	India	1.7	India	1.8	France	1.7	Sweden	2.1
10	Canada	1.6	Switzerland	1.2	Romania	1.7	United Kingdom	2
11	Other	28.5	Other	14.5	Other	15.9	Other	19.6

We also analyzed the ICS (IEC 104) sessions. Figure 10 shows the number of IP addresses establishing a single ICS session, IP addresses with multiple ICS sessions without an HTTP session, and those establishing both HTTP and ICS sessions.

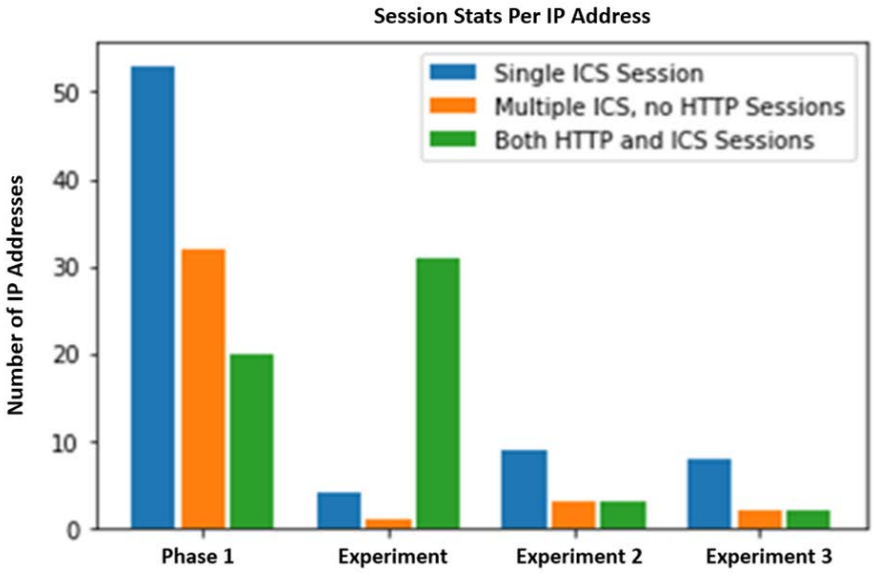

Figure 10. ICS sessions statistics

In Experiment 1, all the ICS traffic from addresses with dual HTTP and ICS sessions was incorrectly formatted IEC 104 messages; they had HTTP format although they were sent to the IEC 104 port 2404. An example is in Figure 11. This appears to be brute-force vulnerability scanning without regard to protocol format.

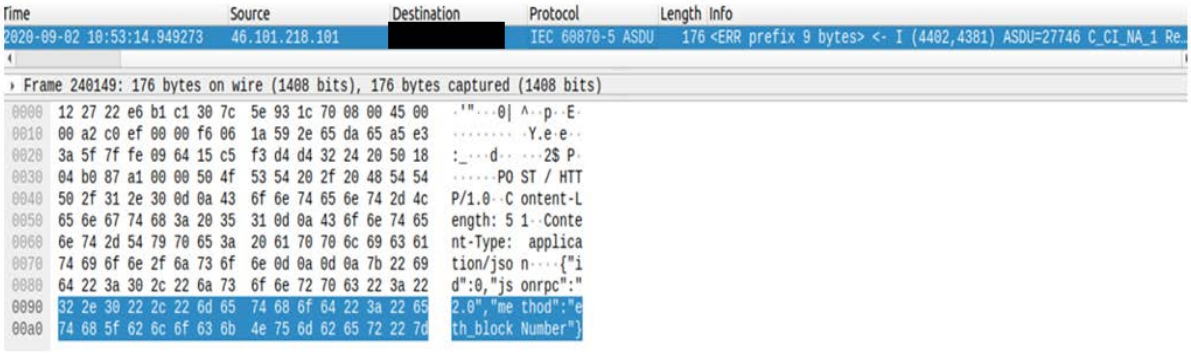

Figure 11. Error traffic sent to the ICS server

Table 4 shows counts of U-format, I-format, error frames, and total IEC traffic. Dougherty Phase 1 and Experiment 1 received more error frames than the other experiments. Table 5 shows daily ICS statistics for total IEC 104 traffic and valid-format IEC 104 traffic. Experiment 1 and Experiment 2 ICS method counts differed, which could be due to SHODAN identifying Experiment 1 as a cloud ICS honeypot early, resulting in increased traffic. The code changes used in Experiments 2 and 3 to make the honeypot less obvious may have actually discouraged ICS interactions.

Table 4. Counts of IEC traffic methods

Experiment	*U-Format Frames*	*I-Format Frames*	*Error Frames*	*Total IEC traffic*
Dougherty	78	19	557	654
Experiment 1	13	4	171	188
Experiment 2	11	4	13	28
Experiment 3	8	2	14	24

Table 5. Daily IEC 104 traffic

	Total IEC 104 Traffic				*Valid IEC 104 Traffic*			
Experiment	Mean	Min	Max	Std. Dev	Mean	Min	Max	Std. Dev
Dougherty	10.532	0	130	27.749	2.341	0	8	1.676
Experiment 1	10.444	0	16	2.733	2.429	0	6	1.678
Experiment 2	2.545	0	6	1.827	3.0	0	4	1.095
Experiment 3	2.667	0	9	2.539	2.0	0	3	0.894

A list of addresses used for these joint HTTP and ICS sessions is given in Table 6. The increase in this kind of traffic over Dougherty's implementation suggests a U.S. cloud deployment is more attractive to indiscriminate attackers. Experiments 2 and 3 did not experience the same increased level as Experiment 1. Table 7 shows the distribution of countries establishing ICS sessions.

Table 6. IP addresses that had both HTTP and ICS sessions with Experiment 1

IP Address	*HTTP Traffic Count*	*ICS Traffic Count*	*Country*	*AS Organization*
46.101.218.101	2	4	Germany	Digital Ocean-ASN
46.101.237.230	1	3	Germany	Digital Ocean-ASN
46.101.242.159	1	6	Germany	Digital Ocean-ASN
46.101.244.9	1	2	Germany	Digital Ocean-ASN
46.101.248.32	11	5	Germany	Digital Ocean-ASN
104.248.130.250	5	11	Germany	Digital Ocean-ASN
104.248.132.111	7	14	Germany	Digital Ocean-ASN
104.248.134.171	11	23	Germany	Digital Ocean-ASN
104.248.142.114	6	10	Germany	Digital Ocean-ASN
104.248.242.91	1	5	Germany	Digital Ocean-ASN
104.248.248.56	5	10	Germany	Digital Ocean-ASN
104.248.252.252	9	19	Germany	Digital Ocean-ASN
122.228.19.79	2	1	China	Wenzhou, Zhejiang Province, P.R. China
134.122.69.254	3	5	Germany	Digital Ocean-ASN
134.122.93.160	5	4	Germany	Digital Ocean-ASN
161.35.144.212	2	4	Netherlands	Digital Ocean-ASN
161.35.149.123	2	4	Netherlands	Digital Ocean-ASN

IP Address	*HTTP Traffic Count*	*ICS Traffic Count*	*Country*	*AS Organization*
161.35.152.122	1	2	Netherlands	Digital Ocean-ASN
161.35.66.65	2	4	Germany	Digital Ocean-ASN
161.35.67.107	2	7	Germany	Digital Ocean-ASN
161.35.69.100	2	4	Germany	Digital Ocean-ASN
161.35.77.119	2	6	Germany	Digital Ocean-ASN
161.35.78.92	1	2	Germany	Digital Ocean-ASN
161.35.82.72	3	6	Germany	Digital Ocean-ASN
161.35.95.234	2	2	Netherlands	Digital Ocean-ASN
161.35.95.69	3	4	Netherlands	Digital Ocean-ASN
167.172.219.157	6	3	United States	Digital Ocean-ASN
202.107.226.2	1	1	China	China-net

Table 7. Distribution of ICS sessions by country

	Dougherty Phase 1		*Experiment 1*		*Experiment 2*		*Experiment 3*	
	Country	Percentage of Total	Country	Percentage of Total	Country	Percentage of Total	Country	Percentage of Total
1	United States	41.9	Germany	74.7	Netherlands	31.6	United States	47.1
2	Germany	25.2	Netherlands	12.7	China	31.6	Russia	17.6
3	United Kingdom	13.7	United States	5.1	Romania	15.8	China	17.6
4	Singapore	6.6	Romania	3.8	Russia	10.5	Romania	11.8
5	China	6.4	China	3.8	United States	10.5	Netherlands	5.9
6	Canada	2.7						
7	Romania	2.0						
8	Russia	1.4						
9	Netherlands	0.7						

B. SIMILARITIES BETWEEN TRAFFIC IN EXPERIMENTS

We used cosine similarity analysis (Dangeti, 2017) to identify distributional similarities between experiments. Cosine similarity measures the cosine of the angle between two vectors to determine how much they are pointing in the same direction on a scale of 0 to 1.

We compared cosine similarity experiments with vectors of four elements (see Table 8). Each vector represented average weekly counts of HTTP GET commands, HTTP POST commands, other HTTP methods (CONNECT, PROPFID, NONE, and HEAD), and all IEC methods including I-format, U-format, and error frames (incorrectly formatted IEC messages). The raw data is in Appendix C. The average cosine similarity on counts for consecutive weeks was 0.715 for Experiment 1, 0.774 for Experiment 2, and 0.994 for Experiment 3. These averages are like standard deviations and can be divided into the differences between experiments to provide a measure of significance (see Table 9). As can be seen, the results of the experiments were not significantly different because usually two standard deviations away from the mean is considered as a minimum for significance.

Table 8. Cosine similarity for total traffic of each experiment

	Dougherty Phase 1	*Experiment 1*	*Experiment 2*	*Experiment 3*
Average cosine similarity of total traffic for all weeks	0.794	0.715	0.774	0.994

Table 9. Measure of significance of total traffic for comparison groups

	Dougherty versus Experiment 1	*Experiment 1 versus Experiment 2*	*Experiment 2 versus Experiment 3*
Measure of significance of average cosine similarity for total traffic	Dougherty - 0.100	Experiment 1 - 0.083	Experiment 2 - 0.284
	Experiment 1 - 0.110	Experiment 2 - 0.076	Experiment 3 - 0.221

We used the same method to get the cosine similarity for just IEC 104 methods. The elements in these vectors were counts of I-format frames, U-format frames, and error frames. The average cosine similarities were 0.996, 0.791, and 0.829 for the three experiments (Table 10). The measures of significance are displayed in Table 11, and raw data is in Appendix C.

Table 10. Cosine similarity for IEC 104 traffic distribution

	Dougherty	*Experiment 1*	*Experiment 2*	*Experiment 3*
Average cosine similarity of IEC 104 traffic for all weeks	0.617	0.996	0.791	0.829

Table 11. Measure of significance of IEC 104 traffic for comparison groups

	Dougherty versus Experiment 1	*Experiment 1 versus Experiment 2*	*Experiment 2 versus Experiment 3*
Measure of significance of average cosine similarity of IEC 104 traffic	Dougherty - 0.614	Experiment 1 - 0.206	Experiment 2 - 0.048
	Experiment 1 - 0.381	Experiment 2 -0.259	Experiment 3 - 0.046

C. BEHAVIORAL ANALYSIS

We examined how attackers interacted with each of our experiments to identify unique behaviors.

1. ASDU Traffic Analysis

We studied valid ASDU traffic (packets consisting of correctly formatted I-frame and U-frame messages) sent to and accepted by the IEC 104 server. Table 12 shows the four IP addresses that sent valid ASDU data frames in Experiment 1. The attacker's ASDU valid interactions in Experiment 1 were consistent: a TCP three-way handshake was established, an activation test U-frame was sent to the IEC 104 server, and the server responded with a confirmation U-frame. The attacker then sent a U-frame message to start data transfer, and then an I-frame interrogation command. The GridPot IEC 104 server responded with data generated by the GridLAB-D simulation, and this sequence continued until the attacker did a three-way TCP sign off. This differed from what Dougherty observed where the attackers never responded with TCP acknowledgement messages and did not end the session with a three-way TCP sign off. All the command payloads received

by the IEC 104 server in Experiment 1 were identical (see Figure 12). This suggests that these may be addresses of Web crawlers like SHODAN searching the Internet for IEC 104 systems.

Table 12. Addresses sending valid ASDU traffic to Experiment 1

IP Address	Country	Domain Name	Number of Sessions
185.142.236.35	Netherlands	wine.census.shodan.io	1
89.248.167.131	Netherlands	mason.census.shodan.io	1
193.37.255.114	Slovakia	ppman.ro	1
167.172.219.157	United States	tab.census.shodan.io	1

```
2020-09-17 00:42:36,834 New IEC104 session from 71.6.167.142 (3d66dbe6-173c-4bbc-a7f7-6ac1f8275fa5)
2020-09-17 00:42:36,834 New IEC 104 connection from 71.6.167.142:53446. (3d66dbe6-173c-4bbc-a7f7-6ac1f8275fa5)
2020-09-17 00:42:36,835 ('71.6.167.142', 53446) ---> u_frame. TESTFR act. (3d66dbe6-173c-4bbc-a7f7-6ac1f8275fa5)
2020-09-17 00:42:36,835 ('71.6.167.142', 53446) <--- u_frame 0x68 0x4 0x83 0x0 0x0 0x0  (3d66dbe6-173c-4bbc-a7f7-6ac1f8275fa5)
2020-09-17 00:42:40,158 ('71.6.167.142', 53446) ---> u_frame. STARTDT act. (3d66dbe6-173c-4bbc-a7f7-6ac1f8275fa5)
2020-09-17 00:42:40,159 ('71.6.167.142', 53446) <--- u_frame 0x68 0x4 0xb 0x0 0x0 0x0  (3d66dbe6-173c-4bbc-a7f7-6ac1f8275fa5)
2020-09-17 00:42:44,278 ('71.6.167.142', 53446) ---> i_frame: 0x68 0xe 0x0 0x0 0x0 0x0 0x64 0x1 0x6 0x0 0xff 0xff 0x0 0x0 0x0 0x0.
(3d66dbe6-173c-4bbc-a7f7-6ac1f8275fa5)
2020-09-17 00:42:44,279 ('71.6.167.142', 53446) <--- i_frame 0x68 0xe 0x0 0x0 0x2 0x0 0x64 0x1 0x7 0x0 0x28 0x1e 0x0 0x0 0x0 0x0
```

Figure 12. Valid ASDU interaction with SHODAN remote host

Tables 13 and 14 show those IP addresses that had valid ASDU interactions with Experiments 2 and 3. The China-based attacker in Experiment 2 established a TCP session, exchanged U-frame messages, sent an I-frame message, received an I-frame message from the IEC 104 server, and then tried to send another I-frame message, which was identified as a spurious retransmission due to having a duplicate ACK of the data frame the host had just received. The behavior of this attacker could be a replay attack (Maynard, Et al., 2020, p. 6) or could be due to TCP connectivity issues.

Table 13. Addresses sending valid ASDU traffic to Experiment 2

IP address	*Country*	*Domain Name*	*Number of Sessions*
106.75.3.52	China	ucloud.cn	2
185.142.236.34	Netherlands	hat.census.shodan.io	1
71.6.167.142	United States	census9.shodan.io	1

Table 14. Valid ASDU interactions with Experiment 3

IP address	*Country*	*Domain Name*	*Number of Sessions*
94.102.49.193	Netherlands	cloud.census.shodan.io	1
185.165.190.34	Russia	no record	1

D. INTERESTING ATTACKS

Figure 13 shows a typical dual session HTTP and IEC 104 message, sent to both the HTTP and IEC 104 servers for Experiment 1. The IP address associated with this attack also sent the same attack to the HTTP port. These attacks try to exploit unprotected JavaScript object remote procedure calls (JSON-RPC) in the Ethereum cryptocurrency network to steal cryptocurrency (Cheng et al., 2019, p.47). This attack is a known Ethereum node probe which tests whether the victim has an insecure HTTP JSON-RPC endpoint.

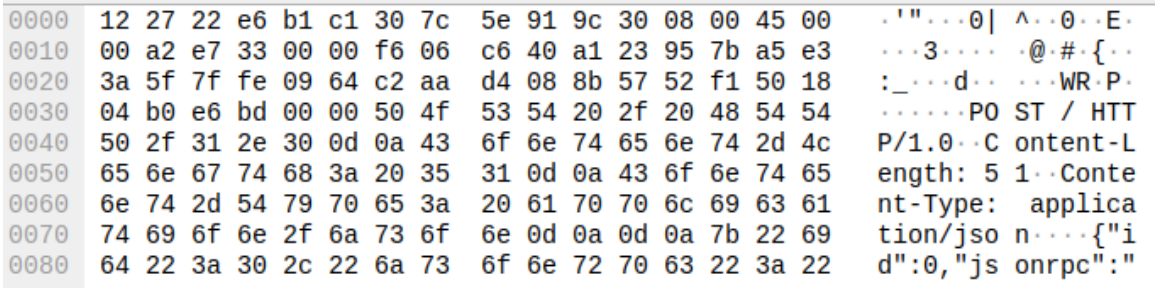

Figure 13. JSON RPC attack

Experiment 1 was indirectly affected by a DNS cache poisoning attack (Lazarevski, n.d.). Traffic to a China-based website was redirected to our HTML page for displaying

the honeypot status. This is likely attributable to DigitalOcean recycling IP addresses, and our URL may have once hosted a site with malicious software that attackers use to infect other machines.

Experiment 3 encountered more HTTP traffic than the other experiments. A single URL accounted for approximately one-third of its traffic, which, according to the result of an NsLookup query, was registered to our Experiment 3 IP address in October 2017 and likely related to a DNS cache poisoning attack (Lazarevski, n.d.). Redirected traffic occurred from the start of Experiment 3, so it is likely that the address was poisoned before being assigned to us by DigitalOcean.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND FUTURE WORK

This thesis explored the cloud deployment of a power-grid honeypot. It changed the deployment environments in three ways to see how they affected actions by adversaries. Our data could make honeypots more believable and better at collecting information on an adversary.

Experiment 1 showed less IEC 104 traffic in the cloud environment than in the physical environment, but similar HTTP traffic volume. This might be due to the rarity of ICS systems in the cloud and disinterest by adversaries in attacking them, but this could change as ICS systems become more common in the cloud. ICS interactions with the cloud environment were more complex than with the physical implementation, suggesting that a cloud GridPot implementation may appear more realistic to adversaries.

Experiment 2 explored the effects of code changes to make the GridPot less obvious. Traffic for Experiment 2 was less than that for Experiment 1, but it is unclear why; perhaps the novelty effect was wearing off since a less obvious honeypot should be more desirable to attack.

Experiment 3 showed more traffic in an Asia-based deployment than a US-based deployment. The rate of interaction in Experiment 3 was significantly higher than Dougherty's Phase 1. However, one-third of the data appears to be due to an existing DNS poisoning attack, so what we did cannot explain all the higher volume. This should be an issue for anyone using a cloud service that recycles Internet addresses. Most data collected was HTTP traffic, showing IEC 104 is less used in the cloud.

We lacked time to test the deployment of Conpot alone, as described in Chapter 4. Using the installation guide we created, which covers both physical Conpot installation (Appendix A) and cloud-based Conpot installation (Appendix B), such deployment should not be difficult. It might be even better to test Conpot in a more general honeypot framework such as T-Pot for both physical and cloud-based deployment.

Time constraints limited the amount of data we could collect. However, the honeypots for Experiments 2 and 3 are still collecting data. That data could be analyzed for

changes in intrusion patterns or adversarial behaviors over time, and for finding patterns over a longer period.

We identified a section of code that could be modified to make GridPot less obvious. We lacked time to implement it because it would require considerable analysis of the linkages between the honeypot and modeling layers of GridPot.

APPENDIX A: PHYSICAL CONPOT INSTALLATION

This appendix contains Chapter 2 of our installation guide, which provides instructions for building a virtualized environment on a physical device and installing Conpot for local implementation. That chapter is included below.

2 Building Conpot in a Local Linux Environment

2.1 Setting Up Local Linux VM:

2.1.1 If on VirtualBox, create a new virtual machine with the following specifications:
- Default CPU settings
- 16 GB RAM (We used 16GB, that much RAM likely unnecessary)
- 50 GB Storage

2.1.2 Do installation using an image of Ubuntu 18.04 with a desktop environment. For the sake of this installation, we used the username "conpot," you may use any username you prefer.

2.1.3 Ensure your VM boots up into a desktop environment before resuming.

2.2 Installing Conpot on Local Linux VM:

2.2.1 Ensure your Ubuntu Installation is up to date:
- **$ sudo apt update**

2.2.2 Install dependencies:
- **$ sudo apt-get install git libsmi2ldbl smistrip libxslt1-dev python3.6-dev libevent-dev default-libmysqlclient-dev**

2.2.3 Install virtualenv:
- **$ sudo apt-get install python3-pip**
- **$ sudo pip3 install virtualenv**

2.2.4 Create a Virtual Environment (conpot is environment name in this case):
- **$ virtualenv --python=python3.6 conpot**

2.2.5 Activate the environment ('conpot' is the environment name in this case):
- **$ source conpot/bin/activate**

2.2.6 Upgrade and install basic tools dependencies within the environment:
- **$ pip install --upgrade pip**
- **$ pip install --upgrade setup tools**
- **$ pip install cffi**

2.2.7 Install base (table) version of Conpot from PYPI:
- **$ pip install conpot**

2.2.8 Test Conpot using the testing configuration- You are will verify that the conpot boots without issues in the testing configuration. Remember that the default and testing templates use inaccurate port numbers (i.e. 8800 for http instead of 80). Installation and use directions for a terminal-based browser called "lynx" is in section 4.5 of this guide:

- **$ conpot -f --template default**

2.2.8.1 Verify Conpot is up:

- Open browser and go to "**127.0.0.1:8800**"
- You should see a minimal Web page with the word "Technodrome" at the top
- Ensure you can see this before moving on
- Use CTR+C to shut down the conpot

2.3 Building a Config File: To proceed and operate using the "Default" template instead of "Test" template you must build a config file and map it to your instance of conpot.

2.3.1 Find the file "testing.cfg" (Should be at \home\conpot\lib\python3.6\site-packages\conpot)

2.3.2 Make a copy of "testing.cfg" and change name to "config.cfg"

2.3.3 Place your new "config.cfg" in a new location (I placed mine at \home\conpot\)

2.3.4 Open "config.cfg" for editing:

- **$ cd ~/conpot/**
- **$ nano config.cfg**

2.3.5 Make the following changes under [Virtual_file_system] change to:

- **data_fs_url = /tmp/**
- **fs_url = tar:///home/conpot/conpot/lib/python3.6/site-packages/conpot/data.tar**
- Ensure a data.tar file is in the above location

2.3.6 Test config.cfg file using default template- You will verify that the conpot boots without issues in the default configuration. Remember that the default and testing templates use inaccurate port numbers (i.e. 8800 for http instead of 80). *NOTE: Installation and use directions for a terminal-based browser called "lynx" is in section 4.5 of this guide*:

- **$ conpot –c ~/conpot/config.cfg --template default**

2.3.6.1 Verify Conpot is up:

- Open browser and go to "**127.0.0.1:8800**"
- You should see a minimal Web page with the word "Technodrome" at the top
- Ensure you can see this before moving on
- Use CTR+C to shut down the conpot

2.4 Install and Configure Authbind: To operate the conpot using the correct port numbers, ports less than 1024 must by bound. However, a normal user is unauthorized to bind ports and conpot will not boot in root. Therefore, Authbind allows for the binding of ports without root.

2.4.1 Install Authbind:

- **$ sudo apt-get install authbind**

2.4.2 Conduct the following steps for the following ports: 21, 69, 80, 102, 161, 502, 623- *NOTE: in step 'b.' below our username was conpot, therefore substitute in your username*:

- **$ sudo touch /etc/authbind/byport/(port#)** (I.e. ~/byport/80)
- **$ sudo chown conpot:conpot /etc/authbind/byport/(port#)**
- **$ sudo chmod 755 /etc/authbind/byport/(port#)**

*NOTE**:** For port numbers greater than 512, additional verification is needed by the system to bind. You must add '!' in front of port 623 in byport folder (IPMI will NOT work w/o this step). You will need 'root' access to make this change. (i.e. /etc/authbind/byport/!623)*

2.5 Correct Port Numbers Within Default Template: Most likely because of binding authorization, the default template uses modified port numbers. Now that you have set up authbind correctly, you may now update your conpot to operate from the correct port numbers.

2.5.1 Navigate to default template folder

2.5.1.1 If using File Manager:

- Go to "/home/conpot/lib/python3.6/site-packages/conpot/templates/default"

2.5.1.2 If using terminal:

- **$ cd "/home/conpot/lib/python3.6/site-packages/conpot/templates/default**

2.5.2 Every protocol has a folder, and in said folder is an .xml file bearing its name. Need not worry about the ssl folders or files.

2.5.2.1 In every .xml file update it to ensure the port number in Line 1 matches the proper port. See Below:

- BACNET: 47808
- ENIP: 44818
- FTP: 21
- HTTP: 80
- IPMI: 623
- MODBUS: 502
- S7COM: 102
- SNMP: 161
- TFTP: 69

2.6 Boot Conpot Using the Default Template and Authbind: *(NOTE: without authbind prefix, you will not have permissions to bind to ports less than 1024) NOTE: Installation and use directions for a terminal-based browser called "lynx" is in section 4.5 of this guide:*

2.6.1 **$ authbind conpot -c ~/conpot/config.cfg --template default**

2.6.1.1 Verify Conpot is up (you are now using the correct port number so just the IP should suffice):

- Open browser and go to "**127.0.0.1**"
- You should see a minimal Web page with the word "Technodrome" at the top
- Ensure you can see this before moving on

2.7 Verify Conpot Visibility Outside Your Home/ Local Network:

2.7.1 Ensure your network is bridged on your Virtual Machine

- For VirtualBox: Shut down your Ubuntu VM and go to Virtualbox
- Right click on your VM and select 'Settings'
- In 'Network' under 'Adapter 1', select 'Bridged Adapter' in 'Attached to:'
- Click 'OK' to save changes and reboot your VM

2.7.2 Verify your outward facing IP- To verify that your conpot is visible outside of your home/local network you must know your IP address. Open a browser in your vm and navigate to http://ip4.me. This will tell you your outward facing IP.

2.7.3 Boot up your conpot- Since that you have ended your previous session you will initialize your virtual environment before initializing your conpot. This will be necessary anytime that you close the terminal window.

- **$ source conpot/bin/activate** (conpot was the name of my virtual environment)
- **$ authbind conpot -c ~/conpot/config.cfg --template default**

2.7.4 On any other device open a browser and navigate to the IP address you acquired in step 2.7(b.). You should see a minimal Web page with the word "Technodrome" at the top. If you do not, verify the following:

- You are using the correct outward facing IP
- You have boot your conpot with prefix 'authbind'
- You have properly configured bridged adapter on your VM
- Your local/home network has the appropriate port forwarding on your internal firewalls to allow for traffic on the ports listed in section 2.5.2.1.

APPENDIX B: CLOUD-BASED CONPOT INSTALLATION

This appendix contains Chapter 3 of our installation guide, which provides instructions for creating a droplet in the DigitalOcean cloud and installing a virtualized Ubuntu 18.04 environment with Conpot for cloud-based use.

3 Building Conpot in a Cloud Environment (DigitalOcean)

3.1 Create a Droplet on DigitalOcean:

3.1.1 Select “Get Started with a Droplet” on your Project Home page
3.1.2 Choose an image (Ubuntu 18.04.3 (LTS) x64)
3.1.3 Choose a plan (Your choice: We chose ‘General Purpose’)
- 16 GB/ 4 CPUs
- 50 GB SSD disk
- 5 TB transfer

3.1.4 Disk options (Your choice: We chose ‘1x SSD’)
3.1.5 Add block storage (Your choice: We did NOT ‘Add Volume’)
3.1.6 Chose a data center region (Your choice: We chose ‘San Francisco’)
3.1.7 Select additional options
3.1.7.1 We selected ‘IPv6’
3.1.7.2 We selected ‘Monitoring’
3.1.7.3 We selected ‘User data’ (add the following code):

```
#!/bin/bash
set -euo pipefail
USERNAME=nps # TODO: Customize the sudo non-root username here
# Create user and immediately expire password to force a change on login
useradd --create-home --shell “/bin/bash” --groups sudo “${USERNAME}”
passwd --delete “${USERNAME}”
chage --lastday 0 “${USERNAME}”
# Create SSH directory for sudo user and move keys over
home_directory=“$(eval echo ~${USERNAME})”
mkdir --parents “${home_directory}/.ssh”
cp /root/.ssh/authorized_keys “${home_directory}/.ssh”
chmod 0700 “${home_directory}/.ssh”
chmod 0600 “${home_directory}/.ssh/authorized_keys”
chown --recursive “${USERNAME}”:”${USERNAME}” “${home_directory}/.ssh”
# Disable root SSH login with password
```

sed --in-place 's/^PermitRootLogin.*/PermitRootLogin prohibit-password/g' /etc/ssh/sshd_config
if sshd -t -q; then systemctl restart sshd fi

3.1.7.4 On Line 4 you can change username ('nps') to what you like

3.1.8 Authentication
- Select 'SSH keys'
- Select 'New SSH Key'
- Create a key using directions in section 4.1
- Copy your key created into box labeled 'SSH key content'
- Name your SSH key in the box labeled 'Name'
- Select 'Add SSH Key'

3.1.9 How many Droplets?
- Your choice (We chose '1 droplet')

3.1.10 Choose a hostname
- Your choice

3.1.11 Add Tags
- Your choice (We left it empty)

3.1.12 Select Project
- Should auto select your account/ project

3.1.13 Add backups
- Your choice (We selected to include backups: does incur additional charges)

3.1.14 Select 'Create Droplet'

3.1.15 Boot and login to Droplet
- (In DigitalOcean) Select 'Droplets' in the left column
- Record the IP address listed for the droplet (you will need this later)
- Select your droplet (click on the droplet name)
- Select 'Access'
- Select' Launch Console' (as per earlier code in section 3.1.7.3, password will be required to change upon login with username)

3.2 Installing Conpot on DigitalOcean:

3.2.1 Ensure your Ubuntu Installation is up to date:
- **$ sudo apt update**

3.2.2 Install dependencies:
- **$ sudo apt-get install git libsmi2ldbl smistrip libxslt1-dev python3.6-dev libevent-dev default-libmysqlclient-dev**

3.2.3 Install virtualenv:
- **$ sudo apt-get install python3-pip**
- **$ sudo pip3 install virtualenv**

3.2.4 Create a Virtual Environment (conpot is environment name in this case)

- **$ virtualenv --python=python3.6 conpot**

3.2.5 Activate the environment ('conpot' is the environment name in this case):
- **$ source conpot/bin/activate**

3.2.6 Upgrade and install basic tools dependencies within the environment:
- **$ pip install --upgrade pip**
- **$ pip install --upgrade setup tools**
- **$ pip install cffi**

3.2.7 Install base (table) version of Conpot from PYPI:
- **$ pip install conpot**

3.2.8 Test Conpot using the testing configuration- You will verify that the conpot boots without issues in the testing configuration. Remember that the default and testing templates use inaccurate port numbers (i.e. 8800 for http instead of 80). *NOTE: Only 1 terminal session can be open while using the console on DigitalOcean. Recommend running conpot as a background process by adding an '&' to the end of the terminal command*:
- **$ conpot -f --template default**

Or
- **$ conpot -f --template default &** (to run as a background process)

3.2.8.1 Verify Conpot is up:
- **$ ps -a** (verify conpot process is running, if ran as background process)
- Using the IP address recorded in section 3.1.15, open local browser and go to "**1.2.3.4:8800**"
  (where 1.2.3.4 is your IP address)
- You should see a minimal Web page with the word "Technodrome" at the top
- Ensure you can see this before moving on

3.2.8.2 Shutdown conpot:
- If conpot was running as a background process:
  $ ps -a
  $ kill -9 <pid listed for conpot from ps -a> (i.e**. $ kill -9 1234**)
- If not run as a background process:
  Use CTR+C to shut down the conpot

3.3 Building a Config File: To proceed and operate using the "Default" template instead of "Test" template you must build a config file and map it to your instance of conpot.

3.3.1 Find the file "testing.cfg" (Should be at \home\conpot\lib\python3.6\site-packages\conpot)

3.3.2 Make a copy of "testing.cfg" and change name to "config.cfg"

3.3.3 Place your new “config.cfg” in a new location (I placed mine at \home\conpot\)

3.3.4 Open “config.cfg” for editing:

- **$ cd ~/conpot/**
- **$ nano config.cfg**

3.3.5 Make the following changes under [Virtual_file_system]:

- **data_fs_url = /tmp/**
- **fs_url = tar:///home/conpot/conpot/lib/python3.6/site-packages/conpot/data.tar**
- NOTE: Ensure a data.tar file is at the above location

3.3.6 Test config.cfg file using default template- You will verify that the conpot boots without issues in the default configuration. *NOTE: Only 1 terminal session can be open while using the console on DigitalOcean. Recommend running conpot as a background process by adding an ‘&’ to the end of the terminal command*.

3.3.7 **$ conpot –c ~/conpot/config.cfg --template default**
Or
$ conpot –c ~/conpot/config.cfg --template default &
(to run as a background process)

3.3.7.1 Verify Conpot is up:

- **$ ps -a** (verify conpot process is running, if ran as background process)
- Using the IP address recorded earlier, open local browser and go to “**1.2.3.4:8800**”
  (where 1.2.3.4 is your IP address)
- You should see a minimal Web page with the word “Technodrome” at the top
- Ensure you can see this before moving on

3.3.7.2 Shutdown conpot:

- If conpot was run as a background process:
  $ ps -a
  $ kill -9 <pid listed for conpot from ps -a> (i.e**. $ kill -9 1234**)
- If not run as a background process:
  Use CTR+C to shut down the conpot (if not run as background process)

3.4 Install and Configure Authbind: To operate the conpot using the correct port numbers, ports less than 1024 must by bound. However, a normal user is unauthorized to bind ports and conpot will not boot in root. Therefore, Authbind allows for the binding of ports without root.

3.4.1 Install Authbind:

- **$ sudo apt-get install authbind**

3.4.2 Conduct the following steps for the following ports: 21, 69, 80, 102, 161, 502, 623- *NOTE: in step 'b.' below our username was conpot, therefore substitute in your username*:

- **$ sudo touch /etc/authbind/byport/(port#)** (I.e. ~/byport/80)
- **$ sudo chown conpot:conpot /etc/authbind/byport/(port#)**
- **$ sudo chmod 755 /etc/authbind/byport/(port#)**

*NOTE**:** For port numbers greater than 512, additional verification is needed by the system to bind. You must add '!' in front of port 623 in byport folder (IPMI will NOT work w/o this step). You will need 'root' access to make this change. (i.e. /etc/authbind/byport/!623)*

3.5 Correct Port Numbers Within Default Template: Most likely because of binding authorization, the default template uses modified port numbers. Now that you have set up authbind correctly, you may now update your Conpot to operate from the correct port numbers.

3.5.1 Navigate to default template folder:

- If using File Manager go to "/home/conpot/lib/python3.6/site-packages/conpot/templates/default"
- If using terminal:
  $ cd "/home/conpot/lib/python3.6/site-packages/conpot/templates/default

3.5.2 Every protocol has a folder, and in said folder is an .xml file bearing its name. Need not worry about the ssl folders or files.

3.5.3 In every .xml file update it to ensure the port number in Line 1 matches the proper port. See Below:

- BACNET: 47808
- ENIP: 44818
- FTP: 21
- HTTP: 80
- IPMI: 623
- MODBUS: 502
- S7COM: 102
- SNMP: 161
- TFTP: 69

3.6 Boot Conpot Using the Default Template and Authbind: (*NOTE: without authbind prefix, you will not have permissions to bind to ports less than 1024) NOTE: Only 1 terminal session can be open while using the console on DigitalOcean. Recommend running conpot as a background process by adding an '&' to the end of the terminal command.*

3.6.1 **$ authbind conpot -c ~/conpot/config.cfg --template default**
or

$ authbind conpot -c ~/conpot/config.cfg --template default &
(to run as a background process)

3.6.2 Verify Conpot is up:

- **$ ps -a** (verify conpot process is running, if ran as background process)
- Using the IP address recorded earlier, open local browser and go to "**1.2.3.4**"
  (where 1.2.3.4 is your IP address)
- You should see a minimal Web page with the word "Technodrome" at the top
- Ensure you can see this before moving on

APPENDIX C: COSINE SIMILARITY DATA

The tables below show the data used to conduct the cosine similarity analysis in Section V.C. These tables contain weekly counts of HTTP and IEC methods for all experiments, values used for the vector elements in cosine similarity, weekly cosine similarity results for HTTP and IEC method distribution, and IEC only method distribution.

Dougherty weekly HTTP and IEC 104 method counts

Week	*CONNECT*	*GET*	*POST*	*OPTIONS*	*HEAD*	*NONE*	*PROPFIND*	*Invalid IEC 104*	*I Frames*	*U Frames*
1	1	235	12	0	4	19	0	0	1	2
2	2	131	49	1	3	11	0	83	1	4
3	3	2075	701	0	3	49	0	1	0	1
4	8	559	779	3	6	84	0	2	3	7
5	8	813	1392	1	3	90	0	2	0	2
6	4	461	776	1	4	49	0	83	1	5
7	5	359	161	4	5	79	0	9	1	4
8	7	807	1510	0	2	31	0	1	1	7
9	1	421	902	1	1	24	0	213	0	4
10	3	208	20	0	1	9	0	1	0	1
11	8	369	53	1	4	55	0	6	1	4
12	2	1288	2405	4	4	139	0	2	0	1
13	5	397	131	0	1	20	0	1	2	5
14	0	24	1	0	0	0	0	13	0	6
15	8	536	735	1	3	0	0	0	3	8
16	1	439	728	0	2	46	0	2	0	1
17	2	852	1576	14	10	105	1	1	2	5
18	0	78	3	0	0	29	0	0	0	0
19	1	141	7	0	0	44	0	131	0	1
20	0	5	0	0	0	2	0	0	0	0
21	0	523	73	0	6	135	0	2	3	7
22	2	899	1465	0	4	83	0	6	0	2

Experiment 1 weekly HTTP and IEC 104 method counts

Week	*CONNECT*	*GET*	*POST*	*OPTIONS*	*HEAD*	*NONE*	*PROPFIND*	*Invalid IEC 104*	*I Frames*	*U Frames*
1	1	3153	64	3	2	78	0	65	3	8
2	6	789	890	0	0	25	0	56	0	2
3	6	295	38	1	5	19	0	33	1	3

Experiment 2 weekly HTTP and IEC 104 method counts										
Week	*CONNECT*	*GET*	*POST*	*OPTIONS*	*HEAD*	*NONE*	*PROPFIND*	*Invalid IEC 104*	*I Frames*	*U Frames*
1	0	529	831	0	2	17	0	2	3	4
2	2	610	43	19	11	162	3	9	1	7
3	2	680	747	1	7	29	0	2	0	0

Experiment 3 weekly HTTP and IEC 104 method counts										
Week	*CONNECT*	*GET*	*POST*	*OPTIONS*	*HEAD*	*NONE*	*PROPFIND*	*Invalid IEC 104*	*I Frames*	*U Frames*
1	0	6913	779	0	16	54	0	1	1	2
2	6	9388	88	4	41	91	0	2	0	3
3	4	9668	1267	1	39	126	0	11	1	3

Average weekly HTTP and IEC 104 method counts										
Experiment	*CONNECT*	*GET*	*POST*	*OPTIONS*	*HEAD*	*NONE*	*PROPFIND*	*Invalid IEC 104*	*I Frames*	*U Frames*
Dougherty	3.227	528.182	612.682	1.409	3	50.136	0.045	25.409	0.864	3.5
Experiment 1	4.333	1412.333	330.667	1.333	2.333	40.667	0	51.333	1.333	4.333
Experiment 2	1.333	606.333	540.333	6.667	6.667	69.333	1	4.333	1.333	3.667
Experiment 3	3.333	8656.333	711.333	1.667	32	90.333	0	4.667	0.667	2.667

Average method counts used for HTTP and IEC 104 cosine similarity comparisons				
Experiment	*GET*	*POST*	*Other Methods*	*IEC*
Dougherty	528.182	612.682	57.818	29.455
Experiment 1	1412.333	330.667	48.667	57
Experiment 2	478	304	81.333	34.667
Experiment 3	8656.333	711.333	127.667	8

Weekly cosine similarity for individual experiments				
Week	*Dougherty*	*Experiment 1*	*Experiment 2*	*Experiment 3*
1	NaN	NaN	NaN	NaN
2	0.815	0.679	0.571	0.995
3	0.845	0.751	0.978	0.993
4	0.81			
5	0.995			
6	0.995			
7	0.808			
8	0.774			
9	0.976			
10	0.499			
11	0.992			
12	0.592			
13	0.725			
14	0.765			
15	0.496			
16	0.995			
17	0.999			
18	0.501			
19	0.746			
20	0.744			
21	0.984			
22	0.627			

Weekly average HTTP and IEC 104 cosine similarity per experiment			
Dougherty	*Experiment 1*	*Experiment 2*	*Experiment 3*
0.794	0.715	0.774	0.994

IEC weekly cosine similarity for individual experiments				
Week	*Dougherty*	*Experiment 1*	*Experiment 2*	*Experiment 3*
1	NaN	NaN	NaN	NaN
2	0.048	0.992	0.795	0.906
3	0.740	0.999	0.786	0.751
4	0.808			
5	0.808			
6	0.748			
7	0.933			
8	0.538			
9	0.158			
10	0.720			
11	0.971			
12	0.983			
13	0.572			
14	0.548			
15	0.392			
16	0.418			
17	0.572			
18	NaN			
19	NaN			
20	NaN			
21	NaN			
22	0.522			

Weekly average IEC cosine similarity per experiment			
Dougherty	*Experiment 1*	*Experiment 2*	*Experiment 3*
0.617	0.996	0.791	0.829

LIST OF REFERENCES

American Public Power Association. (n.d.). *Electricity basics*. Retrieved October 2, 2020, from https://www.publicpower.org/public-power/electricity-basics

Antón, S. D., Fraunholz, D., Lipps, C., Pohl, F., Zimmermann, M., & Schotten, H. D. (2017). Two decades of SCADA exploitation: A brief history. *2017 IEEE Conference on Application, Information and Network Security (AINS)*, 98–104. https://doi.org/10.1109/AINS.2017.8270432

Barak, I. (2020, June 11). Cybereason's newest honeypot shows how multistage ransomware attacks should have critical infrastructure providers on high alert. *Journal of Cyber Policy*. https://journalofcyberpolicy.com/2020/06/12/cybereasons-newest-honeypot-shows-multistage-ransomware-attacks-critical-infrastructure-providers-high-alert/

Beach-Westmoreland, N., & Styczynski, J. (2019). *When the lights went out*. https://www.boozallen.com/content/dam/boozallen/documents/2016/09/ukraine-report-when-the-lights-went-out.pdf

Bokhari, M., Shallal, Q., &Tamandani, Y. (2016). Cloud computing service models: A comparative study. *2016 3rd International Conference on Computing for Sustainable Global Development*, 890–895. https://ieeexplore.ieee.org/document/7724392

Bhutani, A., & Wadhwani, P. (2020, January). *SCADA Market Share 2020–2026. Global Market Insights, Inc*.(Report No.GMI1925) https://www.gminsights.com/industry-analysis/scada-supervisory-control-and-data-acquisition-market

Centre For The Protection of National Infrastructure. (2010). *Configuring and managing remote access for Industrial Control Systems.* https://us-cert.cisa.gov/sites/default/files/recommended_practices/RP_Managing_Remote_Access_S508NC.pdf

Chandra, N., & Madhuri, T. M. (2012). Cloud security using honeypot systems. *International Journal of Scientific & Engineering Research*, 3(3), 1–6.

Cheng, Z., Hou, R., Li, R., Zhou, Y., Luo, X., Li, J., Ren K., (2019). Towards a first step to understand the cryptocurrency stealing attack on ethereum. *USENIX Association, 22nd international Symposium on Research in Attacks, Intrusions and Defenses, 14*, 47–60. http://usenix.org/systm/files/raid2019-cheng.pdf

Combs, L. (2011, December 15). *Cloud computing for SCADA*. Control Engineering. https://www.controleng.com/articles/cloud-computing-for-scada/Conpot. (n.d.). Retrieved July 10, 2020, from https://conpot.readthedocs.io/en/latest/

Dangeti, P. (2017). Statistics for machine learning. Packt Publishing Ltd.

DigitalOcean. (2020a) *Choosing the right droplet plan.* https://www.digitalocean.com/docs/droplets/resources/choose-plan/

DigitalOcean. (2020b). *Projects*. https://www.digitalocean.com/docs/projects/

Dougherty, J. (2020). Evasion of honeypot detection mechanisms through improved interactivity of ICS-based systems. [Master's thesis, Naval Postgraduate School]. NPS Archive: Calhoun. http://hdl.handle.net/10945/66065

Dragos Inc. (2017). *CrashOverride*. https://www.dragos.com/wp-content/uploads/CrashOverride-01.pdf

Dragos Inc. (2020, February 3*). EKANS ransomware and ICS operations*. https://www.dragos.com/blog/industry-news/ekans-ransomware-and-ics-operations/

Dusseault, L. (2007). *HTTP extensions for web distributed authoring and versioning (WebDAV)*[Memorandum]. Internet Engineering Task Force. https://tools.ietf.org/html/rfc4918

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). *Hypertext transfer protocol-HTTP/1.1*.[Memorandum] Internet Engineering Task Force. https://www.w3.org/Protocols/rfc2616/rfc2616.html

Gavin, R. (2018, August 14). *Implementing a cybersecurity strategy for cloud-based SCADA.* Control Engineering. https://www.controleng.com/articles/implementing-a-cybersecurity-strategy-for-cloud-based-scada/

Gjermundrød, H., & Dionysiou, I. (2015). CloudhoneyCY: An integrated honeypot framework for cloud infrastructures. *Proceedings of the 8th International Conference on Utility and Cloud Computing*, 630–635.

Gonzales, D., Kaplan, J. M., Saltzman, E., Winkelman, Z., & Woods, D. (2017). Cloud-trust—A security assessment model for Infrastructure as a Service (IaaS) clouds. *IEEE Transactions on Cloud Computing, 5(3),* 523–536. https://doi.org/10.1109/TCC.2015.2415794

Gridlabd.org (2020). *GridLAB-D*. https://github.com/gridlab-d/gridlab-d

Huston, B. (2013, October 2). *Just a reminder, SIP is a popular scanning target.* State of Security. https://stateofsecurity.com/just-a-reminder-sip-is-a-popular-scanning-target

Hyun, D. (2018). *Collecting cyberattack data for industrial control systems using honeypots* [Master's thesis, Naval Postgraduate School]. NPS Archive: Calhoun. http://hdl.handle.net/10945/58316

Kaspersky. (2020, January 10). *Dustman wiper attack on Bapco oil company*. https://ics-cert.kaspersky.com/news/2020/01/10/bapco-dustman/

Kendrick, M. M., & Rucker, Z. A. (2019). *Energy-grid threat analysis using honeypots* [Master's Thesis, Naval Postgraduate School]. NPS Archive: Calhoun. http://hdl.handle.net/10945/62843

Khoshnaw, R. B.. (2017). A comparative analysis between low and high level honeypots. *Cihan University-Erbil Scientific Journal*, 2017(Special-1), 41–50. https://doi.org/10.24086/cuesj.si.2017.n1a4

Lantero, A. (2014). *How microgrids work.* Energy.Gov. https://www.energy.gov/articles/how-microgrids-work

Lazarevski, B. (n.d.). *Anatomy of a DNS cache poisoning attack* [Presentation]. Online. https://owasp.org/www-chapter-ghana/assets/slides/DNS_Cache_Poisoning(OWASP_GHANA).pdf

Lehrig, S., Eikerling, H., & Becker, S. (2015). Scalability, elasticity, and efficiency in cloud computing: A systematic literature review of definitions and metrics. *2015 11th International ACM SIGSOFT Conference on Quality of Software Architectures*, 83–92. https://doi.org/10.1145/2737182.2737185

Maynard, P., McLaughlin, K. (2020). *Toward understanding Man-on-the-Side attacks (MotS) in SCADA Networks.* Queen's University Belfast, UK. https://arxiv.org/pdf/2004.14334.pdf

Matousek, P. (2017). *Description and analysis of IEC 104 protocol*. (Report No. FIT-TR-2017-12). Brno University of Technology. https://www.fit.vut.cz/research/publication-file/11570/TR-IEC104.pdf

Morsy, M. A., Grundy, J., & Müller, I. (2010). *An analysis of the cloud computing security problem*. Swinburne University of Technology. https://arxiv.org/ftp/arxiv/papers/1609/1609.01107.pdf

National Institute of Standards and Technology. (2013) Cloud computing standards roadmap working group. (Department of Commerce, Washington, D.C.), *Federal Information Processing Standards Publications (FIPS-PUBS) Special Publication SP 500–291r2*. https://doi.org/10.6028/NIST.SP.500-291r2

National Service Center for Environmental Publications. (July, 2012). *Cyber security 101 for water utilities* (EPA 817-K-12-004). Environmental Protection Agency. https://nepis.epa.gov/Exe/ZyNET.exe/P100KL4T.TXT

Piggin, R. (2015). Are industrial control systems ready for the cloud? *International Journal of Critical Infrastructure Protection*, 9, 38–40. https://doi.org/10.1016/j.ijcip.2014.12.005

Provos, N. (2008). *Developments of the Honeyd virtual honeypot*.Honeyd.org Retrieved October 31, 2020, from http://www.honeyd.org/

Quantalytics. (2019). *Q GridPot*. Quantalytics. https://www.quantalytics.com/q-GridPot/

Redwood, O., Lawrence, J., & Burmester, M. (2015). A Symbolic honeynet framework for SCADA system threat intelligence. In M. Rice & S. Shenoi (Eds.), *Critical Infrastructure Protection IX* ,pp. 103–118.

Serbanescu, A. V., Obermeier, S., & Yu, D.-Y. (2015). ICS threat analysis using a large-scale honeynet. *3rd International Symposium for ICS & SCADA Cyber Security Research 2015*,20-30. https://doi.org/10.14236/ewic/ICS2015.3

SHODAN. (n.d.). What is Shodan?. Retrieved October 31, 2020, from https://help.shodan.io/the-basics/what-is-shodan

Sk4ld. (2015). *GridPot* (Version 2) [Computer software]. Github. https://github.com/sk4ld/GridPot

Stouffer, K., Pillitteri, V., Lightman, S., Abrams, M., & Hahn, A. (2015) Guide to Industrial Control Systems (ICS) security. (National Institute of Standards and Technology, Gaithersburg, MD), *NIST Special Publication (SP) 800–82, Rev. 2*. https://doi.org/10.6028/NIST.SP.800-82r2.

Tesfahun, A., & Bhaskari, D. L. (2016). A SCADA Testbed for investigating cyber security vulnerabilities in critical infrastructures. *Automatic Control and Computer Sciences, 50(1),* 54–62. https://doi.org/10.3103/S0146411616010090

Tinankoria, D., & Babak, R. (2017). Cloud computing: A review of the concepts and deployment models. *International Journal of Information Technology and Computer Science, 9(6),* 50–58. https://doi.org/10.5815/ijitcs.2017.06.07U.S.

Wireshark. (n.d.). *Wireshark user's guide: Version 3.5.0*.Wireshark https://www.wireshark.org/download/docs/user-guide.pdf

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California